

EEEEEEEEEEEEEEEE	DDDDDDDDDDDDDD	DD	FFFFFFFFFFFFFFFF
EEEEEEEEEEEEEEEE	DDDDDDDDDDDDDD		FFFFFFFFFFFFFFFF
EEEEEEEEEEEEEEEE	DDDDDDDDDDDDDD		FFFFFFFFFFFFFFFF
EEE	DD	DD	FFF
EEE	DD	DD	FFF
EEE	DD	DD	FFF
EEE	DD	DD	FFF
EEE	DD	DD	FFF
EEE	DD	DD	FFF
EEEEEEEEEEEEEEEE	DD	DD	FFFFFFFFFFFFFFFF
EEEEEEEEEEEEEEEE	DD	DD	FFFFFFFFFFFFFFFF
EEEEEEEEEEEEEEEE	DD	DD	FFFFFFFFFFFFFFFF
EEE	DD	DD	FFF
EEE	DD	DD	FFF
EEE	DD	DD	FFF
EEE	DD	DD	FFF
EEE	DD	DD	FFF
EEEEEEEEEEEEEEEE	DDDDDDDDDDDDDD	DD	FFF
EEEEEEEEEEEEEEEE	DDDDDDDDDDDDDD		FFF
EEEEEEEEEEEEEEEE	DDDDDDDDDDDDDD		FFF

5
Va
--
00
00
00
00
00
00
00
00
00
7F
7F
7F
7F
7F
7F
7F
7F
7F

EEEEEEEEEE	DDDDDDDD	FFFFFFFF	UU	UU	TTTTTTTT	IIIIII	LL	
EEEEEEEEEE	DDDDDDDD	FFFFFFFF	UU	UU	TTTTTTTT	IIIIII	LL	
EE	DD	DD	FF	UU	UU	TT	LL	
EE	DD	DD	FF	UU	UU	TT	LL	
EE	DD	DD	FF	UU	UU	TT	LL	
EE	DD	DD	FF	UU	UU	TT	LL	
EEEEEEEE	DD	DD	FFFFFF	UU	UU	TT	LL	
EEEEEEEE	DD	DD	FFFFFF	UU	UU	TT	LL	
EE	DD	DD	FF	UU	UU	TT	LL	
EE	DD	DD	FF	UU	UU	TT	LL	
EE	DD	DD	FF	UU	UU	TT	LL	
EE	DD	DD	FF	UU	UU	TT	LL	
EE	DD	DD	FF	UU	UU	TT	LL	
EEEEEEEE	DDDDDDDD	FF	UUUUUUUU	UU	TT	IIIIII	LLLLLLLL	...
EEEEEEEE	DDDDDDDD	FF	UUUUUUUU	UU	TT	IIIIII	LLLLLLLL	...

[illegible]

E

[IDENT ('V04-000'),

(++

```
*****
**
**  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
**  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
**  ALL RIGHTS RESERVED.
**
**  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
**  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
**  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
**  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
**  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
**  TRANSFERRED.
**
**  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
**  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
**  CORPORATION.
**
**  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
**  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
**
*****
```

FACILITY: VAX/VMS EDF (EDIT/FDL) UTILITY

ABSTRACT: This facility is used to create, modify, and optimize
FDL specification files.

ENVIRONMENT: NATIVE/USER MODE

AUTHOR: Ken F. Henderson Jr.

CREATION DATE: 27-Mar-1981

MODIFIED BY:

V03-012	RRB0005	Rowland R. Bradley	16 Jan 1984
		Fix EDF\$RESET_SCROLL to avoid overwriting the output generated by a command procedure with SET VERIFY set.	
V03-011	RRB0004	Rowland R. Bradley	13 Jan 1984
		Fix MAX_FACTOR to prevent division by zero.	
V03-010	KFH0010	Ken Henderson	10 Sep 1983
		Support for named UICs.	
V03-009	KFH0009	Ken Henderson	8 Aug 1983
		Changes for sepearte compilation.	
V03-008	KFH0008	Ken Henderson	28 Jul 1983
		Added CALC_REC_OVERHEAD and CALC_BUC_OVERHEAD	

to centralize the arithmetic.
Fixed EDF\$LINE_PARSED to not invert the
order of block comment lines.

V03-007 KFH0007 Ken Henderson 26 Apr 1983

Fix NUMBER_INPUT to set INPUT_VALUE
and INPUT_NUMBER also. Modify
SCAN_DEFINITION routine to accept
FATAL parameter. Modify CURRENT_LT_TEST
and CURRENT_GT_TEST routines to
reverse precedence of SECONDARY
and SECNUM
- and make SEG_TYPE SECNUM = 7.

V03-006 KFH0006 Ken Henderson 14 Apr 1983

Added support for JOURNAL_ENABLED.
Added MAX_FACTOR, DELETE_PRIMARY_SECTION
routines.

V03-005 KFH0005 Ken Henderson 31 Jan 1983

Added XAB\$C_BN8 and XAB\$C_IN8 to
EDF\$LINE_PARSED. And changed the
reference of FDL\$TYPE to FDL3\$TYPE.

V03-004 KFH0004 Ken Henderson 11 Jan 1983

Modified EDF\$RESET_SCROLL to say
"Created:" in reverse video.

V03-003 KFH0003 Ken Henderson 8 Sept 1982

Modified reference to some variables
to fit with database reorganization.
Also, modified call to ASK_RETURN.

V03-002 KFH0002 Ken Henderson 2 April 1982

Modified INSERT_IN_ORDER to not
start at DEF_HEAD if it was already
at the correct place.

V03-001 KFH0001 Ken Henderson 23-Mar-1982

Modified EDF\$RESET_SCROLL to not
reset the scrolling region unless
it has been set.

-- }

EDFUTIL
V04-000

Source Listing

J 14
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (2) Page 3

```
0103 ENVIRONMENT ('LIB$:EDFUTIL'),
0104
0105 INHERIT (
0106
0107   'SYSSLIBRARY:STARLET',
0108   'SHRLIB$:FDLPARDEF',
0109   'LIB$:EDFSDLMSG',
0110   'LIB$:EDFSTRUCT',
0111   'LIB$:EDFCONST',
0112   'LIB$:EDFTYPE',
0113   'LIB$:EDFVAR',
0114   'LIB$:EDFEXTERN',
0115   'LIB$:EDFCHF'
0116
0117 )]
0118
0119 MODULE EDFUTIL (INPUT,OUTPUT);
```

```
0121 { ++
0122
0123 NUM_LEN -- Calculate the field width of an integer.
0124
0125 This routine will return the number of characters an integer will take up
0126 when printed.
0127
0128 CALLING SEQUENCE:
0129
0130 field-width := NUM_LEN (NUMBER);
0131
0132 INPUT PARAMETERS:
0133
0134 NUMBER
0135
0136 IMPLICIT INPUTS:
0137
0138 none
0139
0140 OUTPUT PARAMETERS:
0141
0142 none
0143
0144 IMPLICIT OUTPUTS:
0145
0146 none
0147
0148 ROUTINES CALLED:
0149
0150 none
0151
0152 ROUTINE VALUE:
0153
0154 The field width
0155
0156 SIGNALS:
0157
0158 none
0159
0160 SIDE EFFECTS:
0161
0162 none
0163
0164 -- }
```



```
0166 [ASYNCHRONOUS] FUNCTION NUM_LEN (  
0167     NUMBER : INTEGER  
0168 ) : INTEGER;  
0169  
0170 VAR  
0171     TEST_VAR : INTEGER;  
0172     TEST_LEN : INTEGER;  
0173  
0174 BEGIN  
0175     IF NUMBER = 0 THEN  
0176     { +  
0177       Just plug a width of 1 if the number is 0.  
0178     - }  
0179     NUM_LEN := 1  
0180  
0181 ELSE  
0182 BEGIN  
0183     { +  
0184     Set the function value according to the magnitude of the number.  
0185     - }  
0186     TEST_VAR := 1000000000;  
0187     TEST_LEN := 10;  
0188 REPEAT  
0189     IF ABS (NUMBER) < TEST_VAR THEN  
0190     TEST_LEN := TEST_LEN - 1;  
0191     TEST_VAR := TEST_VAR DIV 10;  
0192 UNTIL ABS (NUMBER) >= TEST_VAR;  
0193     { +  
0194     Allow for a - sign if negative.  
0195     - }  
0196     IF NUMBER < 0 THEN  
0197     TEST_LEN := TEST_LEN + 1;  
0198     { +  
0199     Now stuff the function value.  
0200     - }  
0201     NUM_LEN := TEST_LEN;  
0202 END;  
0203     { IF FALSE NUMBER = 0 }  
0204 END;  
0205     { NUM_LEN }  
0206  
0207  
0208  
0209  
0210  
0211  
0212  
0213  
0214  
0215  
0216  
0217
```

```
0219  ( ++
0220
0221  MAX_FACTOR -- Produce a number that's a multiple of another.
0222
0223  This function will return the number that's a multiple of one of the
0224  arguments, as long as it doesn't go over a maximum.
0225
0226  CALLING SEQUENCE:
0227
0228  NEWVALUE := MAX_FACTOR (BASE,VALUE,MAX);
0229
0230  INPUT PARAMETERS:
0231
0232  BASE
0233  VALUE
0234  MAX
0235
0236  IMPLICIT INPUTS:
0237
0238  none
0239
0240  OUTPUT PARAMETERS:
0241
0242  none
0243
0244  IMPLICIT OUTPUTS:
0245
0246  none
0247
0248  ROUTINES CALLED:
0249
0250  none
0251
0252  ROUTINE VALUE:
0253
0254  The number (greater or equal to VALUE) that's a multiple of BASE,
0255  unless that number would be greater than MAX - in which case it is MAX.
0256
0257  SIGNALS:
0258
0259  none
0260
0261  SIDE EFFECTS:
0262
0263  none
0264
0265  -- }
```


EDFUTIL
V04-000

Source Listing

N 14
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (6)

Page 7

```
0267 FUNCTION MAX_FACTOR (
0268     BASE      : INTEGER;
0269     VALUE     : INTEGER;
0270     MAX       : INTEGER
0271 ) : INTEGER;
0272
0273 VAR
0274     TEMP      : INTEGER;
0275
0276 BEGIN
0277     IF (VALUE < BASE) OR (BASE = 0) THEN
0278         TEMP := BASE
0279     ELSE
0280         BEGIN
0281             TEMP := VALUE DIV BASE;
0282             IF ((VALUE MOD BASE) <> 0) THEN
0283                 TEMP := TEMP + 1;
0284             TEMP := TEMP * BASE;
0285         END;
0286     IF TEMP > MAX THEN
0287         TEMP := MAX;
0288     MAX_FACTOR := TEMP;
0289
0290 END; { MAX_FACTOR }
```

```
0304 { ++
0305
0306 CALC_REC_OVERHEAD -- Do the arithmetic to figure out overheads.
0307
0308 This function will return the RECORD overhead for a given setup.
0309
0310 CALLING SEQUENCE:
0311
0312 RECORD_OVERHEAD := CALC_REC_OVERHEAD (INDEX_LEVEL);
0313
0314 INPUT PARAMETERS:
0315
0316 INDEX_LEVEL
0317
0318 IMPLICIT INPUTS:
0319
0320 none
0321
0322 OUTPUT PARAMETERS:
0323
0324 none
0325
0326 IMPLICIT OUTPUTS:
0327
0328 none
0329
0330 ROUTINES CALLED:
0331
0332 none
0333
0334 ROUTINE VALUE:
0335
0336 The overhead, according to the RMS structure constants.
0337
0338 SIGNALS:
0339
0340 none
0341
0342 SIDE EFFECTS:
0343
0344 none
0345
0346 -- }
```



```
0348 FUNCTION CALC_REC_OVERHEAD (
0349     INDEX_LEVEL : INTEGER
0350 ) : INTEGER;
0351
0352 VAR
0353     RECORD_OVERHEAD : INTEGER;
0354
0355 BEGIN
0356     RECORD_OVERHEAD := 0;
0357
0358     { +
0359     SDR BUCKET
0360     - }
0361     IF (IDATA[EDF$K_ACTIVE_KEY] <> 0) AND (INDEX_LEVEL = 0) THEN
0362         RECORD_OVERHEAD := RECORD_OVERHEAD + IRC$C_SDROVHSZ3 + IRC$C_RRV OVHSZ3;
0363
0364     { +
0365     ACCOUNT FOR KEY COMPRESSION
0366     - }
0367     IF (
0368         (BDATA[EDF$K_KEY_COMP_WANTED] AND (INDEX_LEVEL = 0))
0369         OR
0370         (BDATA[EDF$K_IDX_COMP_WANTED] AND (INDEX_LEVEL <> 0))
0371     ) THEN
0372         RECORD_OVERHEAD := RECORD_OVERHEAD + IRC$C_KEYCMP OVH;
0373
0374     { +
0375     INDEX BUCKETS
0376     - }
0377     IF INDEX_LEVEL <> 0 THEN
0378         RECORD_OVERHEAD := RECORD_OVERHEAD + IRC$C_MAXVBNSZ;
0379
0380     { +
0381     PRIMARY KEY DATA BUCKETS
0382     - }
0383     IF (IDATA[EDF$K_ACTIVE_KEY] = 0) AND (INDEX_LEVEL = 0) THEN
0384         BEGIN
0385             IF VARIABLE_RECORDS THEN
0386                 RECORD_OVERHEAD := RECORD_OVERHEAD + IRC$C_VAROVHSZ3
0387             ELSE
0388                 RECORD_OVERHEAD := RECORD_OVERHEAD + IRC$C_FIXOVHSZ3;
0389             IF BDATA[EDF$K_REC_COMP_WANTED] THEN
0390                 RECORD_OVERHEAD := RECORD_OVERHEAD + IRC$C_DATCMP OVH;
0391             IF BDATA[EDF$K_KEY_COMP_WANTED] OR BDATA[EDF$K_REC_COMP_WANTED] THEN
0392
0393
0394
0395
0396
0397
0398
0399
0400
0401
0402
0403
0404
```

EDFUTIL
V04-000

Source Listing

D 15
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (8) Page 10

```
0405      RECORD_OVERHEAD      := RECORD_OVERHEAD + IDATA[EDF$K_KEY_SIZE];
0406
0407      END;
0408
0409      CALC_REC_OVERHEAD      := RECORD_OVERHEAD;
0410
0411  END;      { CALC_REC_OVERHEAD }
```



```
0413 { ++
0414
0415 CALC_BUC_OVERHEAD -- Do the arithmetic to figure out overheads.
0416
0417 This function will return the BUCKET overhead for a given setup.
0418
0419 CALLING SEQUENCE:
0420
0421 BUCKET_OVERHEAD := CALC_BUC_OVERHEAD (INDEX_LEVEL);
0422
0423 INPUT PARAMETERS:
0424
0425 INDEX_LEVEL
0426
0427 IMPLICIT INPUTS:
0428
0429 none
0430
0431 OUTPUT PARAMETERS:
0432
0433 none
0434
0435 IMPLICIT OUTPUTS:
0436
0437 none
0438
0439 ROUTINES CALLED:
0440
0441 none
0442
0443 ROUTINE VALUE:
0444
0445 The overhead, according to the RMS structure constants.
0446
0447 SIGNALS:
0448
0449 none
0450
0451 SIDE EFFECTS:
0452
0453 none
0454
0455 -- }
```

EDFUTIL
V04-000

Source Listing

F 15
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (10) Page 12

```
0457 FUNCTION CALC_BUC_OVERHEAD (
0458     INDEX_LEVEL : INTEGER
0459 ) : INTEGER;
0460
0461 BEGIN
0462
0463     IF INDEX_LEVEL = 0 THEN
0464         CALC_BUC_OVERHEAD := BKT$C_OVERHDSZ + BKT$C_DATBKTOVH
0465
0466     ELSE
0467         CALC_BUC_OVERHEAD := BKT$C_OVERHDSZ + BKT$C_ENDOVHD;
0468
0469
0470 END; { CALC_BUC_OVERHEAD }
0471
```



```
0473      ( ++
0474
0475      EDF$RESET_SCROLL -- Reset an ANSI terminal's scroll region.
0476
0477      This routine will put the scroll region back to full screen.
0478      It also clears graphics mode.
0479      It is a Global routine, which is called by the exit handler as well.
0480
0481      CALLING SEQUENCE:
0482
0483      EDF$RESET_SCROLL;
0484
0485      INPUT PARAMETERS:
0486
0487      none
0488
0489      IMPLICIT INPUTS:
0490
0491      LINE_ONE
0492      LINES_PER_PAGE
0493
0494      OUTPUT PARAMETERS:
0495
0496      none
0497
0498      IMPLICIT OUTPUTS:
0499
0500      SYS$OUTPUT: - the scroll region is reset, and possibly graphics mode reset
0501
0502      ROUTINES CALLED:
0503
0504      LIB$SET_SCROLL
0505
0506      ROUTINE VALUE:
0507
0508      none
0509
0510      SIGNALS:
0511
0512      none
0513
0514      SIDE EFFECTS:
0515
0516      none
0517
0518      -- }
```

```
0520 [ASYNCHRONOUS,GLOBAL] PROCEDURE EDF$RESET_SCROLL;
0521
0522 BEGIN
0523     IF NOT AUTO_TUNE THEN
0524     BEGIN
0525         { +
0526         Clear graphics mode if this is a Regis device.
0527         - }
0528         IF REGIS THEN
0529         BEGIN
0530             CHFFLAGS := 0;
0531             WRITEV (OUT_LINE, '(27)\\');
0532             LIB$PUT_LINE (OUT_LINE, ONE, CHFFLAGS);
0533         END;
0534         { +
0535         Now make the scroll region from top to bottom - if it was ever set.
0536         - }
0537         IF SCROLLING_SET THEN
0538             LIB$SET_SCROLL (LINE_ONE, LINES_PER_PAGE);
0539     END;
0540     { IF NOT AUTO_TUNE }
0541     { +
0542     Announce that the file has been created.
0543     - }
0544     IF (
0545         (FILE_CREATED)
0546         AND
0547         (RES_OUTPUT_FILENAME_DESC.DSC$W_LENGTH > 0)
0548     ) THEN
0549     BEGIN
0550         CHFFLAGS := SCR$M_REVERSE;
0551         WRITEV (OUT_LINE, CRLF,
0552         RES_OUTPUT_FILENAME_DESC.DSC$A_POINTER^:RES_OUTPUT_FILENAME_DESC.DSC$W_LENGTH,
0553         ' ', LINES_SHOWN:NUM_LEN(LINES_SHOWN), ' lines');
0554         LIB$PUT_LINE (OUT_LINE, ONE, CHFFLAGS);
0555     END;
0556 END; { EDF$RESET_SCROLL }
```



```
0571      ( ++
0572
0573      CLEAR -- Clear a designated area of the screen.
0574
0575      This routine clears a specific area on the screen and leaves the cursor there.
0576      It bypasses screwing up non-CRT terminals.
0577
0578      CALLING SEQUENCE:
0579
0580      CLEAR (DESTINATION);
0581
0582      INPUT PARAMETERS:
0583
0584      DESTINATION
0585
0586      IMPLICIT INPUTS:
0587
0588      PROMPT LINE
0589      LINE_ONE
0590
0591      OUTPUT PARAMETERS:
0592
0593      none
0594
0595      IMPLICIT OUTPUTS:
0596
0597      SYSS$OUTPUT:
0598
0599      ROUTINES CALLED:
0600
0601      LIB$ERASE_PAGE
0602      LIB$ERASE_LINE
0603
0604      ROUTINE VALUE:
0605
0606      none
0607
0608      SIGNALS:
0609
0610
0611      SIDE EFFECTS:
0612
0613      The selected lines on the screen are cleared (unless hardcopy).
0614
0615      -- }
```

```
0617 PROCEDURE CLEAR (
0618     DESTINATION : INTEGER
0619 );
0620
0621 BEGIN
0622
0623     { +
0624     All this stuff affects only video terminals.
0625     - }
0626     IF (
0627         (VIDEO_TERMINAL)
0628         AND
0629         (NOT AUTO_TUNE)
0630     ) THEN
0631     BEGIN
0632
0633         CASE DESTINATION OF
0634
0635             SCREEN :
0636
0637                 BEGIN
0638
0639                     { +
0640                     The following sequence of junk to the screen
0641                     is overkill to make sure the titles don't
0642                     jump around. (interaction of Pascal I/O and
0643                     screen package I/O...)
0644                     - }
0645                     IF REGIS THEN
0646
0647                         WRITELN (''(27)'Pp;S(E);'(27)'\');
0648
0649                         LIB$ERASE_PAGE (LINE_ONE,COL_ONE);
0650                         WRITELN ('');
0651                         LIB$SET_CURSOR (LINE_ONE,COL_ONE);
0652
0653                     END; { SCREEN }
0654
0655             LOWER_AREA :
0656
0657                 BEGIN
0658
0659                     IF REGIS THEN
0660
0661                         BEGIN
0662
0663                             WRITELN (
0664                                 ''(27)'PpP[27,320];V(W(10,S1,E,SL,479)))[+767];'(27)'\');
0665
0666                             LIB$SET_CURSOR (PROMPT_LINE,COL_ONE);
0667
0668                         END
0669
0670                     ELSE
0671
0672                         LIB$ERASE_PAGE (LOWER_LINE,COL_ONE);
0673
```



```
0674
0675      END;
0676
0677      IF_FULL_PROMPT :    IF FULL_PROMPT OR TEMP_FULL_PROMPT THEN
0678
0679          BEGIN
0680
0681              IF TEMP_FULL_PROMPT THEN
0682
0683                  LIB$WAIT (1.3);
0684
0685                  { +
0686                  The following sequence of junk to the screen
0687                  is overkill to make sure the titles don't
0688                  jump around. (interaction of Pascal I/O and
0689                  screen package I/O...)
0690                  - }
0691
0692                  IF REGIS THEN
0693
0694                      WRITELN (''(27)'Pp;S(E);'(27)'\');
0695
0696                      LIB$ERASE_PAGE (LINE_ONE,COL_ONE);
0697                      WRITELN (' ');
0698                      LIB$SET_CURSOR (LINE_ONE,COL_ONE);
0699
0700                  END;    { IF_FULL_PROMPT }
0701
0702          PAUSE :          QUERY (EDF$K_RETURN);
0703
0704      OTHERWISE
0705
0706          { NULL-STATEMENT } ;
0707
0708      END;    { CASE }
0709
0710      END;    { IF VIDEO_TERMINAL AND NOT AUTO_TUNE }
0711
0712      END;    { CLEAR }
```

```
0714 { ++
0715
0716 CVT_QUAD_DESC -- Routine to convert a quadword to a descriptor.
0717
0718 This routine will take 2 longword arguments and stuff them into a descriptor.
0719
0720 CALLING SEQUENCE:
0721
0722 DESCRIPTOR_VAR := CVT_QUAD_DESC (LONG1, LONG2);
0723
0724 INPUT PARAMETERS:
0725
0726 LONG1
0727 LONG2
0728
0729 IMPLICIT INPUTS:
0730
0731 none
0732
0733 OUTPUT PARAMETERS:
0734
0735 none
0736
0737 IMPLICIT OUTPUTS:
0738
0739 none
0740
0741 ROUTINES CALLED:
0742
0743 none
0744
0745 ROUTINE VALUE:
0746
0747 DESCRIPTOR_VAR
0748
0749 SIGNALS:
0750
0751 none
0752
0753 SIDE EFFECTS:
0754
0755 none
0756
0757 -- }
```


EDFUTIL
V04-000

Source Listing

M 15
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (16) Page 19

```
[ASYNCHRONOUS] FUNCTION CVT_QUAD_DESC (LONG1, LONG2 : LONG) : DESCRIPTOR;
BEGIN
  WITH QUAD_DESC DO
    BEGIN
      { +
      Select the quadword type variant and stuff it.
      - }
      QWHICH                := QWORD;
      TWOLONG.L1             := LONG1;
      TWOLONG.L2             := LONG2;

      { +
      Now select the descriptor type variant and get it.
      - }
      QWHICH                := DWORD;
      CVT_QUAD_DESC          := DSC;
    END;
  END;
END; { CVT_QUAD_DESC }
```

```
0784      { ++
0785
0786      SCAN_DEFINITION -- Search for area and key primaries in the current definition
0787      and log them.
0788
0789      CALLING SEQUENCE:
0790
0791      SCAN_DEFINITION (FATAL);
0792
0793      INPUT PARAMETERS:
0794
0795      FATAL
0796
0797      IMPLICIT INPUTS:
0798
0799      DEF_CURRENT
0800
0801      OUTPUT PARAMETERS:
0802
0803      none
0804
0805      IMPLICIT OUTPUTS:
0806
0807      LOW_AREA
0808      HIGH_AREA
0809      LOW_KEY
0810      HIGH_KEY
0811      FOUND_O
0812      FOUND_AREA
0813      FOUND_KEY
0814
0815      ROUTINES CALLED:
0816
0817      none
0818
0819      ROUTINE VALUE:
0820
0821      none
0822
0823      SIGNALS:
0824
0825      none
0826
0827      SIDE EFFECTS:
0828
0829      none
0830
0831      -- }
```



```
0833 PROCEDURE SCAN_DEFINITION (FATAL : BOOLEAN);
0834
0835 BEGIN
0836
0837   { +
0838   Find out the range of existing keys (assume contiguous).
0839   - }
0840   DEF_CURRENT      := DEF_HEAD;
0841   FOUND_0          := FALSE;
0842   FOUND_AREA       := FALSE;
0843   FOUND_KEY        := FALSE;
0844   LOW_AREA         := 0;
0845   HIGH_AREA        := 0;
0846   LOW_KEY          := 0;
0847   HIGH_KEY         := 0;
0848
0849   REPEAT
0850
0851     WITH DEF_CURRENT^ DO
0852
0853       BEGIN
0854
0855         IF (
0856           (OBJECT_TYPE = PRI)
0857           AND
0858           (PRIMARY = KEY)
0859         ) THEN
0860
0861           BEGIN
0862
0863             IF PRINUM = 0 THEN
0864
0865               FOUND_0      := TRUE;
0866
0867               FOUND_KEY    := TRUE;
0868
0869               IF PRINUM < LOW_KEY THEN
0870
0871                 LOW_KEY    := PRINUM;
0872
0873               IF PRINUM > HIGH_KEY THEN
0874
0875                 HIGH_KEY   := PRINUM;
0876
0877             END;
0878
0879             IF (
0880               (OBJECT_TYPE = PRI)
0881               AND
0882               (PRIMARY = AREA)
0883             ) THEN
0884
0885               BEGIN
0886
0887                 FOUND_AREA  := TRUE;
0888
0889                 IF PRINUM < LOW_AREA THEN
```



```
0890
0891         LOW_AREA      := PRINUM;
0892
0893         IF PRINUM > HIGH_AREA THEN
0894
0895             HIGH_AREA   := PRINUM;
0896
0897         END;
0898
0899         END;      { WITH DEF_CURRENT^ DO }
0900
0901         DEF_CURRENT     := DEF_CURRENT^.FORE;
0902
0903     UNTIL DEF_CURRENT = NIL;
0904
0905     IF (
0906         ((FATAL) OR (HIGH_KEY <> 0))
0907     AND
0908         (NOT FOUND_0)
0909     ) THEN
0910
0911     BEGIN
0912
0913         WRITELN (SHIFT,ANSI_REVERSE,
0914             ' There is no Primary Key in the Current Definition. ',
0915             ANSI_RESET);
0916
0917         IF AUTO_TUNE THEN
0918
0919             LIB$STOP (EDF$_INSFANL,0,0,0)
0920
0921         ELSE
0922
0923         BEGIN
0924
0925             LIB$WAIT (3.0);
0926             LIB$SIGNAL (EDF$_CTRLZ,0,0,0);
0927
0928         END;
0929
0930     END;
0931
0932 END;      { SCAN_DEFINITION }
```



```
0934 { ++
0935
0936 PARSE_INPUT -- Routine to parse input string.
0937
0938 This routine will look at the chosen LIB$TPARSE table.
0939
0940 CALLING SEQUENCE:
0941
0942 PARSE_INPUT (KEY_TABLE_PTR,STATE_TABLE_PTR,DEFAULT_OK,DEFAULT_VALUE);
0943
0944 INPUT PARAMETERS:
0945
0946 KEY_TABLE_PTR
0947 STATE_TABLE_PTR
0948
0949 IMPLICIT INPUTS:
0950
0951 none
0952
0953 OUTPUT PARAMETERS:
0954
0955 none
0956
0957 IMPLICIT OUTPUTS:
0958
0959 INPUT_VALUE
0960
0961 ROUTINES CALLED:
0962
0963 none
0964
0965 ROUTINE VALUE:
0966
0967 none
0968
0969 SIGNALS:
0970
0971 none
0972
0973 SIDE EFFECTS:
0974
0975 none
0976
0977 -- }
```



```
0979 PROCEDURE PARSE_INPUT (
0980     KEY_TABLE      : INTEGER;
0981     STATE_TABLE    : INTEGER;
0982     DEFAULT_OK     : BOOLEAN;
0983     DEFAULT_VALUE  : INTEGER
0984 );
0985
0986 BEGIN
0987     { +
0988     Get the input from the terminal.
0989     - }
0990     INPUT_DESC := NULL_STRING;
0991
0992     { +
0993     If auto answers are enabled and this question has a default - use it.
0994     - }
0995     IF (
0996     (
0997     (TAKE_DEFAULTS)
0998     AND
0999     (IDATA[EDF$K_RESPONSES] = EDF$K_AUTO)
1000     AND
1001     (DEFAULT_OK)
1002     AND
1003     (NOT (QTAB_OFFSET = EDF$K_RETURN))
1004     )
1005     OR
1006     (AUTO_TUNE)
1007     ) THEN
1008
1009     BEGIN
1010
1011         IF NOT AUTO_TUNE THEN
1012
1013             BEGIN
1014
1015                 WRITELN (CRLF);
1016                 LIB$WAIT (0.7);
1017
1018             END;
1019
1020         END
1021
1022     ELSE
1023
1024         BEGIN
1025
1026             IF EOF (INPUT) THEN
1027
1028                 BEGIN
1029
1030                     RESET (INPUT);
1031                     LIB$SIGNAL (EDF$_CTRLZ,0,0,0);
1032
1033                 END;
1034
1035         END;
```



```
1036     READLN (INPUT_STRING);
1037     WRITELN (CRLF);
1038     STR$TRIM (INPUT_DESC, INPUT_STRING);
1039     STR$UPCASE (INPUT_DESC, INPUT_DESC);
1040
1041     END;
1042
1043     { +
1044     If we're journaling our input, save a copy of it to the
1045     journal file.
1046     - }
1047     IF JOURNAL_ENABLED THEN
1048         IF INPUT_DESC.DSC$W_LENGTH > 0 THEN
1049             WRITELN (
1050                 JOURNAL_FILE,
1051                 INPUT_DESC.DSC$A_POINTER^:INPUT_DESC.DSC$W_LENGTH
1052             )
1053         ELSE
1054             WRITELN (JOURNAL_FILE);
1055
1056     { +
1057     See if the answer was defaulted, and if it's allowed to be.
1058     - }
1059     IF INPUT_DESC.DSC$W_LENGTH = 0 THEN
1060         IF DEFAULT_OK THEN
1061             INPUT_VALUE      := DEFAULT_VALUE
1062         ELSE
1063             BEGIN
1064                 LIB$SIGNAL (EDF$_NODEFAULT, 0, 0, 0);
1065             END
1066         ELSE
1067             BEGIN
1068                 { +
1069                 See if it's valid and get it's value.
1070                 - }
1071                 PARAM_BLOCK.TPAS$L_STRINGPTR      := INPUT_DESC.DSC$A_POINTER::UNSIGNED;
1072                 PARAM_BLOCK.TPAS$L_STRINGCNT      := INPUT_DESC.DSC$W_LENGTH;
1073                 ISTATUS := LIB$TPARSE (
1074                     PARAM_BLOCK,
1075                     STATE_TABLE,
1076                     KEY_TABLE
1077                 );
1078             END
1079         END
1080     END
1081
1082     { +
1083     See if it's valid and get it's value.
1084     - }
1085     PARAM_BLOCK.TPAS$L_STRINGPTR      := INPUT_DESC.DSC$A_POINTER::UNSIGNED;
1086     PARAM_BLOCK.TPAS$L_STRINGCNT      := INPUT_DESC.DSC$W_LENGTH;
1087     ISTATUS := LIB$TPARSE (
1088         PARAM_BLOCK,
1089         STATE_TABLE,
1090         KEY_TABLE
1091     );
1092
```

```
1093 INPUT_VALUE      := PARAM_BLOCK.TPASL_PARAM::LONG;  
1094 INPUT_NUMBER     := PARAM_BLOCK.TPASL_NUMBER::LONG;  
1095  
1096 { +  
1097 Even Istatus (low bit clear) means failure.  
1098 - }  
1099 IF NOT ODD (ISTATUS) THEN  
1100 BEGIN  
1101  
1102     IF PARAM_BLOCK.TPASV_AMBIG THEN  
1103         LIB$SIGNAL (EDF$_AMBIG,0,0,0)  
1104     ELSE  
1105         LIB$SIGNAL (EDF$_BADSYNTAX,0,0,0);  
1106  
1107     END;  
1108  
1109 END;  
1110  
1111 { IF NOT INPUT_DESC.DSC$_LENGTH = 0 }  
1112  
1113 END;  
1114 { PARSE_INPUT }  
1115
```



```
1117 ( ++
1118
1119 NUMBER_INPUT -- Routine to get a number from the input string.
1120
1121 This routine will return the integer typed.
1122
1123 CALLING SEQUENCE:
1124
1125 NUMBER_INPUT (NUM_VALUE,DEFAULT_OK,DEFAULT_VALUE);
1126
1127 INPUT PARAMETERS:
1128
1129 none
1130
1131 IMPLICIT INPUTS:
1132
1133 none
1134
1135 OUTPUT PARAMETERS:
1136
1137 NUM_VALUE
1138
1139 IMPLICIT OUTPUTS:
1140
1141 none
1142
1143 ROUTINES CALLED:
1144
1145 none
1146
1147 ROUTINE VALUE:
1148
1149 none
1150
1151 SIGNALS:
1152
1153 none
1154
1155 SIDE EFFECTS:
1156
1157 none
1158
1159 -- }
```

```
1161 PROCEDURE NUMBER_INPUT (
1162     VAR NUM_VALUE : INTEGER;
1163     DEFAULT_OK     : BOOLEAN;
1164     DEFAULT_VALUE  : INTEGER
1165 );
1166
1167 BEGIN
1168     { +
1169     Get the input from the terminal.
1170     - }
1171     INPUT_DESC := NULL_STRING;
1172
1173     { +
1174     If auto answers are enabled and this question has a default - use it.
1175     - }
1176     IF (
1177     (
1178     (TAKE_DEFAULTS)
1179     AND
1180     (IDATA[EDF$K_RESPONSES] = EDF$K_AUTO)
1181     AND
1182     (DEFAULT_OK)
1183     )
1184     OR
1185     (AUTO_TUNE)
1186     ) THEN
1187     BEGIN
1188         IF NOT AUTO_TUNE THEN
1189             BEGIN
1190                 WRITELN (CRLF);
1191                 LIB$WAIT (0.7);
1192             END;
1193         END
1194     ELSE
1195     BEGIN
1196         IF EOF (INPUT) THEN
1197             BEGIN
1198                 RESET (INPUT);
1199                 LIB$SIGNAL (EDF$_CTRLZ,0,0,0);
1200             END;
1201         READLN (INPUT_STRING);
1202         WRITELN (CRLF);
1203         STR$TRIM (INPUT_DESC,INPUT_STRING);
1204     END;
```



```
1218 STR$UPCASE (INPUT_DESC,INPUT_DESC);
1219 PARAM_BLOCK.TP$T_TOKENPTR := INPUT_DESC.DSC$A_POINTER::UNSIGNED;
1220 PARAM_BLOCK.TP$T_TOKENCNT := INPUT_DESC.DSC$W_LENGTH;
1221
1222 END;
1223
1224 { +
1225 If we're journaling our input, save a copy of it to the
1226 journal file.
1227 - }
1228 IF JOURNAL_ENABLED THEN
1229     IF INPUT_DESC.DSC$W_LENGTH > 0 THEN
1230         WRITELN (
1231             JOURNAL_FILE,
1232             INPUT_DESC.DSC$A_POINTER^:INPUT_DESC.DSC$W_LENGTH
1233         )
1234     ELSE
1235         WRITELN (JOURNAL_FILE);
1236
1237 { +
1238 See if the answer was defaulted, and if it's allowed to be.
1239 - }
1240 IF INPUT_DESC.DSC$W_LENGTH = 0 THEN
1241     IF DEFAULT_OK THEN
1242         NUM_VALUE := DEFAULT_VALUE
1243     ELSE
1244         BEGIN
1245             LIB$SIGNAL (EDF$_NODEFAULT,0,0,0);
1246         END
1247 ELSE
1248     BEGIN
1249         { +
1250         Convert it to an integer.
1251         - }
1252         ISTATUS := OT$SCVT_TI_L (INPUT_DESC,NUM_VALUE);
1253         { +
1254         Even Istatus (low bit clear) means failure.
1255         - }
1256         IF NOT ODD (ISTATUS) THEN
1257             BEGIN
1258                 LIB$SIGNAL (EDF$_BADSYNTAX,0,0,0);
```

EDFUTIL
V04-000

Source Listing

K 16
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (22) Page 30

```
1275
1276      END;
1277
1278      END;      { IF NOT INPUT_DESC.DSC$W_LENGTH = 0 }
1279
1280      INPUT_VALUE      := NUM_VALUE;
1281      INPUT_NUMBER      := NUM_VALUE;
1282
1283      END;      { NUMBER_INPUT }
```



```
1285 { ++
1286
1287 MAKE_SCRATCH -- Create a new peice of dynamic memory and init it.
1288
1289 This routine creates a new Line_object, and inits its various fields.
1290
1291 CALLING SEQUENCE:
1292
1293 MAKE_SCRATCH;
1294
1295 INPUT PARAMETERS:
1296
1297 none
1298
1299 IMPLICIT INPUTS:
1300
1301 LINE_OBJECT_TEMPLATE
1302
1303 OUTPUT PARAMETERS:
1304
1305 DEF_SCRATCH
1306
1307 IMPLICIT OUTPUTS:
1308
1309 none
1310
1311 ROUTINES CALLED:
1312
1313 none
1314
1315 ROUTINE VALUE:
1316
1317 none
1318
1319 SIGNALS:
1320
1321 none
1322
1323 SIDE EFFECTS:
1324
1325 none
1326
1327 -- }
```

EDFUTIL
V04-000

Source Listing

M 16
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (24) Page 32

```
1329 [ASYNCHRONOUS] PROCEDURE MAKE_SCRATCH;  
1330  
1331 BEGIN  
1332  
1333     { +  
1334     Allocate some dynamic memory.  
1335     - }  
1336     NEW (DEF_SCRATCH);  
1337  
1338     { +  
1339     Copy over the template.  
1340     - }  
1341     DEF_SCRATCH^      := LINE_OBJECT_TEMPLATE;  
1342  
1343 END;    { MAKE_SCRATCH }
```



```
1345 ( **
1346
1347 CURRENT_GT_TEST -- Compare def_current with test.
1348
1349 This function has the boolean value of the whether or not DEF_CURRENT is
1350 greater than TEST.
1351
1352 CALLING SEQUENCE:
1353
1354 test-val      := CURRENT_GT_TEST (TEST,SCOPE);
1355
1356 INPUT PARAMETERS:
1357
1358 SCOPE
1359
1360 IMPLICIT INPUTS:
1361
1362 DEF_CURRENT
1363
1364 OUTPUT PARAMETERS:
1365
1366 none
1367
1368 IMPLICIT OUTPUTS:
1369
1370 none
1371
1372 ROUTINES CALLED:
1373
1374 none
1375
1376 ROUTINE VALUE:
1377
1378 True if DEF_CURRENT > TEST, false if not.
1379
1380 SIGNALS:
1381
1382 none
1383
1384 SIDE EFFECTS:
1385
1386 none
1387
1388 -- }
```



```
1390 [ASYNCHRONOUS] FUNCTION CURRENT_GT_TEST (
1391                                     TEST
1392                                     EXACT COMPARISON : LINE OBJECT;
1393                                     ) : BOOLEAN;      : BOOLEAN
1394
1395 BEGIN
1396     CURRENT_GT_TEST := FALSE;
1397
1398     { +
1399     Just do a boolean assignment.
1400     - }
1401     IF EXACT_COMPARISON THEN
1402     BEGIN
1403         IF
1404             (PRI_SEQ[DEF_CURRENT^.PRIMARY] > PRI_SEQ[TEST.PRIMARY])
1405         THEN
1406             CURRENT_GT_TEST := TRUE;
1407
1408         IF
1409             (PRI_SEQ[DEF_CURRENT^.PRIMARY] = PRI_SEQ[TEST.PRIMARY])
1410             AND
1411             (DEF_CURRENT^.PRINUM > TEST.PRINUM)
1412         THEN
1413             CURRENT_GT_TEST := TRUE;
1414
1415         IF
1416             (PRI_SEQ[DEF_CURRENT^.PRIMARY] = PRI_SEQ[TEST.PRIMARY])
1417             AND
1418             (DEF_CURRENT^.PRINUM = TEST.PRINUM)
1419             AND
1420             (DEF_CURRENT^.SECNUM > TEST.SECNUM)
1421         THEN
1422             CURRENT_GT_TEST := TRUE;
1423
1424         IF
1425             (PRI_SEQ[DEF_CURRENT^.PRIMARY] = PRI_SEQ[TEST.PRIMARY])
1426             AND
1427             (DEF_CURRENT^.PRINUM = TEST.PRINUM)
1428             AND
1429             (DEF_CURRENT^.SECNUM = TEST.SECNUM)
1430             AND
1431             (DEF_CURRENT^.SECONDARY > TEST.SECONDARY)
1432         THEN
1433             CURRENT_GT_TEST := TRUE;
1434
1435     END
1436 ELSE
1437 BEGIN
```



```
1447
1448
1449     IF (
1450     PRI_SEQ[DEF_CURRENT^.PRIMARY] > PRI_SEQ[TEST.PRIMARY])
1451     THEN
1452         CURRENT_GT_TEST      := TRUE;
1453
1454     IF
1455     (PRI_SEQ[DEF_CURRENT^.PRIMARY] = PRI_SEQ[TEST.PRIMARY])
1456     AND
1457     (DEF_CURRENT^.PRINUM > TEST.PRINUM)
1458     THEN
1459         CURRENT_GT_TEST      := TRUE;
1460
1461     END;
1462
1463 END;    ( CURRENT_GT_TEST )
1464
```



```
1466 ( ++
1467
1468 CURRENT_LT_TEST -- Compare def_current and test.
1469
1470 This function has the boolean value of the whether or not TEST greater than
1471 DEF_CURRENT.
1472
1473 CALLING SEQUENCE:
1474
1475 test-val      := CURRENT_LT_TEST (TEST,SCOPE);
1476
1477 INPUT PARAMETERS:
1478
1479 TEST
1480
1481 IMPLICIT INPUTS:
1482
1483 DEF_CURRENT
1484
1485 OUTPUT PARAMETERS:
1486
1487 none
1488
1489 IMPLICIT OUTPUTS:
1490
1491 none
1492
1493 ROUTINES CALLED:
1494
1495 none
1496
1497 ROUTINE VALUE:
1498
1499 True if DEF_CURRENT < TEST, false if not.
1500
1501 SIGNALS:
1502
1503 none
1504
1505 SIDE EFFECTS:
1506
1507 none
1508
1509 -- }
```



```
1511 [ASYNCHRONOUS] FUNCTION CURRENT_LT_TEST (
1512                                     TEST
1513                                     EXACT_COMPARISON : LINE OBJECT;
1514                                     ) : BOOLEAN;
1515
1516 BEGIN
1517     CURRENT_LT_TEST := FALSE;
1518
1519     { +
1520     Just do a boolean assignment.
1521     - }
1522     IF EXACT_COMPARISON THEN
1523     BEGIN
1524         IF (
1525             PRI_SEQ[DEF_CURRENT^.PRIMARY] < PRI_SEQ[TEST.PRIMARY])
1526         THEN
1527             CURRENT_LT_TEST := TRUE;
1528
1529         IF
1530             (PRI_SEQ[DEF_CURRENT^.PRIMARY] = PRI_SEQ[TEST.PRIMARY])
1531             AND
1532             (DEF_CURRENT^.PRINUM < TEST.PRINUM)
1533         THEN
1534             CURRENT_LT_TEST := TRUE;
1535
1536         IF
1537             (PRI_SEQ[DEF_CURRENT^.PRIMARY] = PRI_SEQ[TEST.PRIMARY])
1538             AND
1539             (DEF_CURRENT^.PRINUM = TEST.PRINUM)
1540             AND
1541             (DEF_CURRENT^.SECNUM < TEST.SECNUM)
1542         THEN
1543             CURRENT_LT_TEST := TRUE;
1544
1545         IF
1546             (PRI_SEQ[DEF_CURRENT^.PRIMARY] = PRI_SEQ[TEST.PRIMARY])
1547             AND
1548             (DEF_CURRENT^.PRINUM = TEST.PRINUM)
1549             AND
1550             (DEF_CURRENT^.SECNUM = TEST.SECNUM)
1551             AND
1552             (DEF_CURRENT^.SECONDARY < TEST.SECONDARY)
1553         THEN
1554             CURRENT_LT_TEST := TRUE;
1555
1556     END
1557     ELSE
1558     BEGIN
```



```
1568
1569
1570     IF (
1571     PRI_SEQ[DEF_CURRENT^.PRIMARY] < PRI_SEQ[TEST.PRIMARY])
1572     THEN
1573         CURRENT_LT_TEST      := TRUE;
1574
1575     IF
1576     (PRI_SEQ[DEF_CURRENT^.PRIMARY] = PRI_SEQ[TEST.PRIMARY])
1577     AND
1578     (DEF_CURRENT^.PRINUM < TEST.PRINUM)
1579     THEN
1580         CURRENT_LT_TEST      := TRUE;
1581
1582     END;
1583
1584 END;    { CURRENT_LT_TEST }
1585
```



```
1587 { ++
1588
1589 CURRENT_EQ_TEST -- Compare def_current and test.
1590
1591 This function has the boolean value of the whether or not TEST is the
1592 same as DEF_CURRENT.
1593
1594 CALLING SEQUENCE:
1595
1596 test-val      := CURRENT_EQ_TEST (TEST,SCOPE);
1597
1598 INPUT PARAMETERS:
1599
1600 SCOPE
1601
1602 IMPLICIT INPUTS:
1603
1604 DEF_CURRENT
1605
1606 OUTPUT PARAMETERS:
1607
1608 none
1609
1610 IMPLICIT OUTPUTS:
1611
1612 none
1613
1614 ROUTINES CALLED:
1615
1616 none
1617
1618 ROUTINE VALUE:
1619
1620 True if DEF_CURRENT = TEST, false if not.
1621
1622 SIGNALS:
1623
1624 none
1625
1626 SIDE EFFECTS:
1627
1628 none
1629
1630 -- }
```

```
1632 [ASYNCHRONOUS] FUNCTION CURRENT_EQ_TEST (
1633     TEST
1634     EXACT_COMPARISON : LINE_OBJECT;
1635     ) : BOOLEAN;
1636
1637 BEGIN
1638
1639     { +
1640     Just do a boolean assignment.
1641     - }
1642     IF EXACT_COMPARISON THEN
1643
1644         CURRENT_EQ_TEST := (
1645             (TEST.OBJECT_TYPE = DEF_CURRENT^.OBJECT_TYPE)
1646             AND
1647             (TEST.PRIMARY = DEF_CURRENT^.PRIMARY)
1648             AND
1649             (TEST.PRINUM = DEF_CURRENT^.PRINUM)
1650             AND
1651             (TEST.SECONDARY = DEF_CURRENT^.SECONDARY)
1652             AND
1653             (TEST.SECNUM = DEF_CURRENT^.SECNUM)
1654         )
1655
1656     ELSE
1657
1658         CURRENT_EQ_TEST := (
1659             (TEST.PRIMARY = DEF_CURRENT^.PRIMARY)
1660             AND
1661             (TEST.PRINUM = DEF_CURRENT^.PRINUM)
1662         );
1663
1664 END; { CURRENT_EQ_TEST }
```



```
1670 { ++
1671
1672 INSERT_BEFORE_CURRENT -- Link the DEF_SCRATCH line_object into the list.
1673
1674 This routine adds DEF_SCRATCH into the list just before DEF_CURRENT.
1675
1676 CALLING SEQUENCE:
1677
1678 INSERT_BEFORE_CURRENT;
1679
1680 INPUT PARAMETERS:
1681
1682 none
1683
1684 IMPLICIT INPUTS:
1685
1686 DEF_SCRATCH
1687 DEF_CURRENT
1688 DEF_HEAD
1689
1690 OUTPUT PARAMETERS:
1691
1692 none
1693
1694 IMPLICIT OUTPUTS:
1695
1696 none
1697
1698 ROUTINES CALLED:
1699
1700 none
1701
1702 ROUTINE VALUE:
1703
1704 none
1705
1706 SIGNALS:
1707
1708 none
1709
1710 SIDE EFFECTS:
1711
1712 none
1713
1714 -- }
```

```
1716 [ASYNCHRONOUS] PROCEDURE INSERT_BEFORE_CURRENT;  
1717  
1718 BEGIN  
1719  
1720 { +  
1721 Make it the new head, if we're adding it before the old head.  
1722 - }  
1723 IF DEF_CURRENT = DEF_HEAD THEN  
1724     DEF_HEAD      := DEF_SCRATCH;  
1725  
1726 { +  
1727 Update the fore and back pointers.  
1728 - }  
1729 DEF_PRED      := DEF_CURRENT^.BACK;  
1730 DEF_SCRATCH^.FORE := DEF_CURRENT;  
1731 DEF_SCRATCH^.BACK := DEF_PRED;  
1732  
1733 IF DEF_PRED <> NIL THEN  
1734     DEF_PRED^.FORE := DEF_SCRATCH;  
1735  
1736 DEF_CURRENT^.BACK := DEF_SCRATCH;  
1737  
1738 { +  
1739 Leave looking at the just inserted line_object.  
1740 - }  
1741 DEF_CURRENT      := DEF_SCRATCH;  
1742  
1743 END; { INSERT_BEFORE_CURRENT }
```



```
1747 { ++
1748
1749 INSERT_AT_CURRENT -- Link the DEF_SCRATCH line_object into the list.
1750
1751 This routine adds DEF_SCRATCH into the list at DEF_CURRENT.
1752
1753 CALLING SEQUENCE:
1754
1755 INSERT_AT_CURRENT;
1756
1757 INPUT PARAMETERS:
1758
1759 none
1760
1761 IMPLICIT INPUTS:
1762
1763 DEF_SCRATCH
1764 DEF_CURRENT
1765 DEF_HEAD
1766
1767 OUTPUT PARAMETERS:
1768
1769 none
1770
1771 IMPLICIT OUTPUTS:
1772
1773 none
1774
1775 ROUTINES CALLED:
1776
1777 none
1778
1779 ROUTINE VALUE:
1780
1781 none
1782
1783 SIGNALS:
1784
1785 none
1786
1787 SIDE EFFECTS:
1788
1789 none
1790
1791 -- }
```

```
1793 [ASYNCHRONOUS] PROCEDURE INSERT_AT_CURRENT;
1794
1795 BEGIN
1796     { +
1797     Make new head or tail, if we're replacing this one.
1798     - }
1800     IF DEF_CURRENT = DEF_HEAD THEN
1801         DEF_HEAD      := DEF_SCRATCH;
1802
1803     IF DEF_CURRENT = DEF_TAIL THEN
1804         DEF_TAIL      := DEF_SCRATCH;
1805
1806     { +
1807     Substitute the links to def_current with links to def_scratch.
1808     - }
1809     DEF_PRED          := DEF_CURRENT^.BACK;
1810     DEF_SUCC          := DEF_CURRENT^.FORE;
1811     DEF_SCRATCH^.FORE := DEF_CURRENT^.FORE;
1812     DEF_SCRATCH^.BACK := DEF_CURRENT^.BACK;
1813
1814     IF DEF_PRED <> NIL THEN
1815         DEF_PRED^.FORE := DEF_SCRATCH;
1816
1817     IF DEF_SUCC <> NIL THEN
1818         DEF_SUCC^.BACK := DEF_SCRATCH;
1819
1820     { +
1821     Get rid of the old def_current, and point def_current to the new king.
1822     - }
1823     DISPOSE (DEF_CURRENT);
1824
1825     DEF_CURRENT      := DEF_SCRATCH;
1826
1827 END; { INSERT_AT_CURRENT }
```



```
1833 ( ++
1834
1835 INSERT_AFTER_CURRENT -- Link the DEF_SCRATCH line_object into the list.
1836
1837 This routine adds DEF_SCRATCH to the list after DEF_CURRENT.
1838
1839 CALLING SEQUENCE:
1840
1841 INSERT_AFTER_CURRENT;
1842
1843 INPUT PARAMETERS:
1844
1845 none
1846
1847 IMPLICIT INPUTS:
1848
1849 DEF_CURRENT
1850 DEF_SCRATCH
1851 DEF_TAIL
1852
1853 OUTPUT PARAMETERS:
1854
1855 none
1856
1857 IMPLICIT OUTPUTS:
1858
1859 none
1860
1861 ROUTINES CALLED:
1862
1863 none
1864
1865 ROUTINE VALUE:
1866
1867 none
1868
1869 SIGNALS:
1870
1871 none
1872
1873 SIDE EFFECTS:
1874
1875 none
1876
1877 -- }
```

```

[ASYNCHRONOUS] PROCEDURE INSERT_AFTER_CURRENT;
BEGIN
    ( +
    Make it the new tail if we're adding it after t
    - )
    IF DEF_CURRENT = DEF_TAIL THEN
        DEF_TAIL      := DEF_SCRATCH;
    ( +
    Update the fore and back pointers.
    - )
    DEF_SUCC          := DEF_CURRENT^.FORE;
    DEF_SCRATCH^.FORE := DEF_SUCC;
    DEF_SCRATCH^.BACK := DEF_CURRENT;
    DEF_CURRENT^.FORE := DEF_SCRATCH;
    IF DEF_SUCC <> NIL THEN
        DEF_SUCC^.BACK := DEF_SCRATCH;
    ( +
    Leave looking at the just inserted line_object.
    - )
    DEF_CURRENT      := DEF_SCRATCH;
END;    { INSERT_AFTER_CURRENT }

```


EDFUTIL
V04-000

Source Listing

C 2
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (37) Page 47

```
1909      ( ++
1910
1911      INCR_CURRENT -- Bump DEF_CURRENT pointer one.
1912
1913      This routine makes DEF_CURRENT point to the next line_object in the list,
1914      as long as it won't fall off the end.
1915
1916      CALLING SEQUENCE:
1917
1918      INCR_CURRENT;
1919
1920      INPUT PARAMETERS:
1921
1922      none
1923
1924      IMPLICIT INPUTS:
1925
1926      DEF_CURRENT
1927
1928      OUTPUT PARAMETERS:
1929
1930      none
1931
1932      IMPLICIT OUTPUTS:
1933
1934      none
1935
1936      ROUTINES CALLED:
1937
1938      LIB$SIGNAL
1939
1940      ROUTINE VALUE:
1941
1942      none
1943
1944      SIGNALS:
1945
1946
1947      SIDE EFFECTS:
1948
1949      none
1950
1951      -- }
```

EDF
V04

244
244
244
244
244
244
244
244
244
245
245
245
245
245
245
245
245
245
245
246
246
246
246
246
246
246

Source Listing

VAX-11 Pascal V2.4-277 Page 48
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (38)

1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964

```

[ASYNCHRONOUS] PROCEDURE INCR_CURRENT;
BEGIN
    { +
      .FORE points to the next line_object in the list.
      - }
    IF DEF_CURRENT <> NIL THEN
        DEF_CURRENT      := DEF_CURRENT^.FORE;
END;    { INCR_CURRENT }

```



```
1966      ( ++
1967
1968      DECR_CURRENT -- Bump back the DEF_CURRENT pointer one.
1969
1970      This routine points DEF_CURRENT to the previous line_object in the list,
1971      as long as it won't run off the end.
1972
1973      CALLING SEQUENCE:
1974
1975      DECR_CURRENT;
1976
1977      INPUT PARAMETERS:
1978
1979      none
1980
1981      IMPLICIT INPUTS:
1982
1983      DEF_CURRENT
1984
1985      OUTPUT PARAMETERS:
1986
1987      none
1988
1989      IMPLICIT OUTPUTS:
1990
1991      none
1992
1993      ROUTINES CALLED:
1994
1995      LIB$SIGNAL
1996
1997      ROUTINE VALUE:
1998
1999      none
2000
2001      SIGNALS:
2002
2003
2004      SIDE EFFECTS:
2005
2006      none
2007
2008      -- }
```

EDFUTIL
V04-000

Source Listing

F 2
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (40) Page 50

2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

```
[ASYNCHRONOUS] PROCEDURE DECR_CURRENT;  
BEGIN  
    { +  
    .BACK points to the previous line_object in the list.  
    - }  
    IF DEF_CURRENT <> NIL THEN  
        DEF_CURRENT      := DEF_CURRENT^.BACK;  
END;    { DECR_CURRENT }
```



```

2023 { ++
2024
2025 NEW_IDENT_LINE -- Make a new Ident and stuff it into the definition.
2026
2027 This routine gets the date and time from the system and makes a new Ident
2028 Line_object, and puts it into the definition linked list.
2029
2030 IT ASSUMES THAT THE LIST IS COMPLETELY EMPTY!!!
2031
2032 CALLING SEQUENCE:
2033
2034 NEW_IDENT_LINE;
2035
2036 INPUT PARAMETERS:
2037
2038 none
2039
2040 IMPLICIT INPUTS:
2041
2042 DEF_SCRATCH
2043 IDENT_STRING
2044
2045 OUTPUT PARAMETERS:
2046
2047 none
2048
2049 IMPLICIT OUTPUTS:
2050
2051 DEF_SCRATCH
2052 DEF_HEAD
2053 DEF_TAIL
2054 DEF_CURRENT
2055
2056 ROUTINES CALLED:
2057
2058 LIB$DATE_TIME
2059 MAKE_SCRATCH
2060
2061 ROUTINE VALUE:
2062
2063 none
2064
2065 SIGNALS:
2066
2067 none
2068
2069 SIDE EFFECTS:
2070
2071 none
2072
2073 -- }

```

```
2075  PROCEDURE NEW_IDENT_LINE;
2076
2077  VAR
2078      DATE_STRING : STRING20;
2079      I           : INTEGER;
2080
2081  BEGIN
2082
2083      { +
2084      Create a place for the Ident to go.
2085      - }
2086      MAKE_SCRATCH;
2087
2088      { +
2089      Get system date and time to put into IDENT line.
2090      - }
2091      LIB$DATE_TIME (DATE_STRING);
2092
2093      { +
2094      Now, copy it into the ident string.
2095      - }
2096      FOR I := 1 TO 20 DO
2097          IDENT_STRING[I] := DATE_STRING[I];
2098
2099      { +
2100      Put an IDENT primary at the head of the linked list.
2101      - }
2102      WITH DEF_SCRATCH^ DO
2103      BEGIN
2104
2105          TEMP_DESCRIPTOR := NULL_STRING;
2106          NEW (TEMP_DESCRIPTOR.DSC$A_POINTER);
2107
2108          TEMP_DESCRIPTOR.DSC$W_LENGTH := IDENT_STRING_LENGTH;
2109          OBJECT_TYPE := PRI;
2110          PRIMARY := IDENT;
2111
2112          FOR I := 1 TO IDENT_STRING_LENGTH DO
2113              TEMP_DESCRIPTOR.DSC$A_POINTER^[I] := IDENT_STRING[I];
2114
2115          LIB$COPY_DXDX (TEMP_DESCRIPTOR, STRING);
2116          DISPOSE (TEMP_DESCRIPTOR.DSC$A_POINTER);
2117
2118      END;      { WITH DEF_SCRATCH^ }
2119
2120      { +
2121      Make the just created line_object the head (and only) one
2122      - }
2123      DEF_CURRENT := DEF_SCRATCH;
2124      DEF_HEAD := DEF_SCRATCH;
2125      DEF_TAIL := DEF_SCRATCH;
2126
2127  END;      { NEW_IDENT_LINE }
```



```
2132 { ++
2133
2134 DELETE_CURRENT -- Unlink DEF_CURRENT from the list and kill it.
2135
2136 This routine removes the line_object pointed to by DEF_CURRENT from the list.
2137
2138 CALLING SEQUENCE:
2139
2140 DELETE_CURRENT;
2141
2142 INPUT PARAMETERS:
2143
2144 none
2145
2146 IMPLICIT INPUTS:
2147
2148 DEF_CURRENT
2149 DEF_TAIL
2150 DEF_HEAD
2151
2152 OUTPUT PARAMETERS:
2153
2154 none
2155
2156 IMPLICIT OUTPUTS:
2157
2158 DEF_CURRENT
2159 DEF_TAIL
2160 DEF_HEAD
2161
2162 ROUTINES CALLED:
2163
2164 EXTRACT_CURRENT
2165
2166 ROUTINE VALUE:
2167
2168 none
2169
2170 SIGNALS:
2171
2172 none
2173
2174 SIDE EFFECTS:
2175
2176 none
2177
2178 -- }
```

```

2180 PROCEDURE DELETE_CURRENT;
2181
2182 BEGIN
2183
2184     IF DEF_CURRENT^.PRIMARY = TITLE THEN
2185     BEGIN
2186
2187         { +
2188         TITLE is always the very 1st line_object in the list.
2189         - }
2190         IF DEF_CURRENT^.FORE = NIL THEN
2191         BEGIN
2192
2193             DISPOSE (DEF_CURRENT);
2194             NEW_IDENT_LINE;
2195
2196         END      { IF TRUE DEF_CURRENT^.FORE = NIL }
2197
2198     ELSE
2199     BEGIN
2200
2201         DEF_HEAD                := DEF_CURRENT^.FORE;
2202         DEF_HEAD^.BACK          := NIL;
2203         DISPOSE (DEF_CURRENT);
2204         DEF_CURRENT              := DEF_HEAD;
2205
2206     END;      { IF FALSE DEF_CURRENT^.FORE = NIL }
2207
2208 END      { IF TRUE DEF_CURRENT^.PRIMARY = TITLE }
2209
2210 ELSE
2211 BEGIN
2212
2213     { +
2214     Make new tail, if we're deleting old tail.
2215     - }
2216     IF (DEF_CURRENT <> NIL) AND (DEF_CURRENT = DEF_TAIL) THEN
2217     BEGIN
2218         DEF_TAIL                := DEF_CURRENT^.BACK;
2219
2220     { +
2221     Make new head, if we're deleting old head.
2222     - }
2223     IF (DEF_CURRENT <> NIL) AND (DEF_CURRENT = DEF_HEAD) THEN
2224     BEGIN
2225         DEF_HEAD                := DEF_CURRENT^.FORE;
2226
2227     { +
2228     Update fore and back pointers.
2229     - }
2230     DEF_PRED                    := DEF_CURRENT^.BACK;
2231     DEF_SUCC                    := DEF_CURRENT^.FORE;

```



```

2237 IF DEF_PRED <> NIL THEN
2238
2239     DEF_PRED^.FORE := DEF_SUCC;
2240
2241 IF DEF_SUCC <> NIL THEN
2242
2243     DEF_SUCC^.BACK := DEF_PRED;
2244
2245 WITH DEF_CURRENT^ DO
2246
2247 BEGIN
2248
2249     IF STRING.DSC$W_LENGTH > 0 THEN
2250
2251         STR$FREE1_DX (STRING);
2252
2253     IF COMMENT.DSC$W_LENGTH > 0 THEN
2254
2255         STR$FREE1_DX (COMMENT);
2256
2257 END;
2258
2259 DISPOSE (DEF_CURRENT);
2260
2261 IF DEF_SUCC <> NIL THEN
2262
2263     DEF_CURRENT := DEF_SUCC
2264
2265 ELSE IF DEF_PRED <> NIL THEN
2266
2267     DEF_CURRENT := DEF_PRED
2268
2269 ELSE
2270
2271     NEW_IDENT_LINE;
2272
2273 END; { IF FALSE DEF_CURRENT^.PRIMARY = TITLE }
2274
2275 END; { DELETE_CURRENT }

```

```
2277 { ++
2278
2279 DELETE_PRIMARY_SECTION -- Get rid of a whole primary section.
2280
2281 This routine removes all the line_objects of particular primary from the
2282 definition linked list.
2283
2284 CALLING SEQUENCE:
2285
2286 DELETE_PRIMARY_SECTION (PRIMARY,PRINUM);
2287
2288 INPUT PARAMETERS:
2289
2290 PRIMARY
2291 PRINUM
2292
2293 IMPLICIT INPUTS:
2294
2295 DEF_CURRENT
2296 DEF_TAIL
2297 DEF_HEAD
2298
2299 OUTPUT PARAMETERS:
2300
2301 none
2302
2303 IMPLICIT OUTPUTS:
2304
2305 DEF_CURRENT
2306 DEF_TAIL
2307 DEF_HEAD
2308
2309 ROUTINES CALLED:
2310
2311 EXTRACT_CURRENT
2312
2313 ROUTINE VALUE:
2314
2315 none
2316
2317 SIGNALS:
2318
2319 none
2320
2321 SIDE EFFECTS:
2322
2323 none
2324
2325 -- }
```



```
2327 PROCEDURE DELETE_PRIMARY_SECTION (WHICHPRIMARY : PRIMARY_TYPE;  
2328                                     WHICHPRINUM : INTEGER);  
2329  
2330 VAR  
2331     DOING      : BOOLEAN;  
2332     DONE       : BOOLEAN;  
2333  
2334 BEGIN  
2335     DEF_CURRENT := DEF_HEAD;  
2336     DOING       := FALSE;  
2337     DONE        := FALSE;  
2338  
2339     { +  
2340     Cycle until we've deleted the section or we're at the end of the list.  
2341     - }  
2342     REPEAT  
2343     { +  
2344     If this is the start of the right primary, flag it.  
2345     - }  
2346     IF (  
2347     (DEF_CURRENT^.OBJECT_TYPE = PRI)  
2348     AND  
2349     (DEF_CURRENT^.PRIMARY = WHICHPRIMARY)  
2350     AND  
2351     (DEF_CURRENT^.PRINUM = WHICHPRINUM)  
2352     ) THEN  
2353     DOING      := TRUE;  
2354  
2355     { +  
2356     If we're in the right primary, delete the sucker!  
2357     - }  
2358     IF DOING THEN  
2359     DELETE_CURRENT  
2360  
2361     ELSE  
2362     { +  
2363     Move on to the next line_object in the list.  
2364     - }  
2365     INCR_CURRENT;  
2366  
2367     { +  
2368     If we're not already off the end, see if this is still the  
2369     right primary. If not, flag that we're done.  
2370     - }  
2371     IF DEF_CURRENT <> NIL THEN  
2372     IF (  
2373     (DOING)  
2374     AND  
2375     (DEF_CURRENT^.OBJECT_TYPE = PRI)  
2376     AND  
2377     (
```

2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394

```

        (DEF_CURRENT^.PRIMARY <> WHICHPRIMARY)
      OR
      (DEF_CURRENT^.PRINUM <> WHICHPRINUM)
    ) THEN
      DONE      := TRUE;
  UNTIL (DONE) OR (DEF_CURRENT = NIL);
END;    { DELETE_PRIMARY_SECTION }

```

N 2
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277 Page 58
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (46)


```
2396      ( ++
2397
2398      INIT_DEF -- Clear out the definition and setup for a new one.
2399
2400      This routine makes room to put a brand new definition in the linked list.
2401
2402      CALLING SEQUENCE:
2403
2404      INIT_DEF;
2405
2406      INPUT PARAMETERS:
2407
2408      none
2409
2410      IMPLICIT INPUTS:
2411
2412      none
2413
2414      OUTPUT PARAMETERS:
2415
2416      none
2417
2418      IMPLICIT OUTPUTS:
2419
2420      DEF_CURRENT
2421      DEF_HEAD
2422
2423      ROUTINES CALLED:
2424
2425      none
2426
2427      ROUTINE VALUE:
2428
2429      none
2430
2431      SIGNALS:
2432
2433      none
2434
2435      SIDE EFFECTS:
2436
2437      none
2438
2439      -- )
```



```
2441  PROCEDURE INIT_DEF;  
2442  
2443  BEGIN  
2444  
2445      { +  
2446      Clear out the list starting at the beginning (if not already empty).  
2447      - }  
2448      DEF_CURRENT := DEF_HEAD;  
2449  
2450      IF DEF_CURRENT <> NIL THEN  
2451      BEGIN  
2452          REPEAT  
2453              DELETE_CURRENT;  
2454          UNTIL DEF_HEAD = DEF_TAIL;  
2455          IF DEF_CURRENT <> NIL THEN  
2456              DELETE_CURRENT;  
2457          END;  
2458      END;  
2459  
2460  END;  
2461  { INIT_DEF }
```



```
2468 ( ++
2469
2470 INSERT_IN_ORDER -- Put DEF_SCRATCH into the list in proper order.
2471
2472 This routine places the line object pointed to by def_scratch in the definition
2473 linked list in its proper place.
2474
2475 CALLING SEQUENCE:
2476
2477 INSERT_IN_ORDER (COLLISION_ACTION);
2478
2479 INPUT PARAMETERS:
2480
2481 COLLISION_ACTION
2482
2483 IMPLICIT INPUTS:
2484
2485 DEF_CURRENT
2486 DEF_TAIL
2487 DEF_HEAD
2488 DEF_SCRATCH
2489
2490 OUTPUT PARAMETERS:
2491
2492 none
2493
2494 IMPLICIT OUTPUTS:
2495
2496 none
2497
2498 ROUTINES CALLED:
2499
2500 none
2501
2502 ROUTINE VALUE:
2503
2504 none
2505
2506 SIGNALS:
2507
2508 none
2509
2510 SIDE EFFECTS:
2511
2512 none
2513
2514 -- }
```

```
2516 [ASYNCHRONOUS] PROCEDURE INSERT_IN_ORDER (
2517         COLCISTON_ACTION : INTEGER
2518     );
2519
2520 VAR
2521     BACKUP_WORKED      : BOOLEAN;
2522
2523 BEGIN
2524     { +
2525     1st, a little conditioning.
2526     - }
2527     IF (
2528     (DEF_SCRATCH^.OBJECT_TYPE = PRI)
2529     AND
2530     (DEF_SCRATCH^.PRIMARY <> TITLE)
2531     ) THEN
2532
2533         DEF_SCRATCH^.STRING.DSC$W_LENGTH      := 0;
2534
2535     DEF_SCRATCH^.FORE      := NIL;
2536     DEF_SCRATCH^.BACK      := NIL;
2537
2538     { +
2539     Now, find the proper place. Start looking at the previous line_object.
2540     - }
2541     BACKUP_WORKED      := FALSE;
2542
2543     IF DEF_CURRENT <> NIL THEN
2544         IF DEF_CURRENT^.BACK <> NIL THEN
2545             BEGIN
2546                 DECR_CURRENT;
2547
2548                 WHILE NOT (
2549                     (CURRENT_GT_TEST(DEF_SCRATCH^,TRUE))
2550                     OR
2551                     (CURRENT_EQ_TEST(DEF_SCRATCH^,TRUE))
2552                     OR
2553                     (DEF_CURRENT^.FORE = NIL)
2554                 ) DO
2555
2556                     INCR_CURRENT;
2557
2558                     BACKUP_WORKED      := (
2559                         (
2560                             (CURRENT_LT_TEST(DEF_SCRATCH^,TRUE))
2561                             AND
2562                             (DEF_CURRENT^.FORE = NIL)
2563                         )
2564                         OR
2565                         (CURRENT_EQ_TEST(DEF_SCRATCH^,TRUE))
2566                     );
2567
2568                     END; { IF DEF_CURRENT^.BACK <> NIL }
```



```
2573 IF NOT BACKUP_WORKED THEN
2574 BEGIN
2575     { +
2576     The quick look didn't work, now scan the entire list.
2577     - }
2578     DEF_CURRENT := DEF_HEAD;
2579     WHILE NOT (
2580       (CURRENT_GT_TEST(DEF_SCRATCH^,TRUE))
2581       OR
2582       (CURRENT_EQ_TEST(DEF_SCRATCH^,TRUE))
2583       OR
2584       (DEF_CURRENT^.FORE = NIL)
2585     ) DO
2586       INCR_CURRENT;
2587 END; { IF NOT BACKUP_WORKED }
2588 { +
2589 Now insert it according to how it was (found).
2590 - }
2591 IF CURRENT_GT_TEST(DEF_SCRATCH^,TRUE) THEN
2592   INSERT_BEFORE_CURRENT
2593 ELSE IF CURRENT_EQ_TEST(DEF_SCRATCH^,TRUE) THEN
2594   BEGIN
2595     IF COLLISION_ACTION = REPLACE_OBJ THEN
2596       INSERT_AT_CURRENT
2597     ELSE IF COLLISION_ACTION = AFTER_OBJ THEN
2598       INSERT_AFTER_CURRENT;
2599     { IF COLLISION_ACTION = IGNORE_OBJ THEN 'NULL-STATEMENT' }
2600   END
2601 ELSE IF DEF_CURRENT^.FORE = NIL THEN
2602   BEGIN
2603     DEF_TAIL := DEF_CURRENT;
2604     INSERT_AFTER_CURRENT;
2605   END;
2606 END; { INSERT_IN_ORDER }
```



```
2629 ( ++
2630
2631 FIND_OBJECT -- Locate a line_object in the definition list.
2632
2633 This function returns with DEF_CURRENT pointing to the desired line
2634 object - if it finds it, in which case it's function value is true.
2635
2636 CALLING SEQUENCE:
2637
2638 BOOLEAN_VAR := FIND_OBJECT (LINE_OBJECT_TYPE,PRIMARY,PRINUM,SECONDARY,SECNUM);
2639
2640 INPUT PARAMETERS:
2641
2642 OBJECT_TYPE
2643 PRIMARY
2644 PRINUM
2645 SECONDARY
2646 SECNUM
2647
2648 IMPLICIT INPUTS:
2649
2650 none
2651
2652 OUTPUT PARAMETERS:
2653
2654 none
2655
2656 IMPLICIT OUTPUTS:
2657
2658 DEF_CURRENT
2659
2660 ROUTINES CALLED:
2661
2662 none
2663
2664 ROUTINE VALUE:
2665
2666 TRUE/FALSE DEPENDING ON FOUND STATUS
2667
2668 SIGNALS:
2669
2670 none
2671
2672 SIDE EFFECTS:
2673
2674 none
2675
2676 -- }
```



```
2678 FUNCTION FIND_OBJECT (
2679     OBJ_TYP      : LINE_OBJECT_TYPE;
2680     PRIM         : PRIMARY_TYPE;
2681     PRIMNUM      : INTEGER;
2682     SECO         : SECONDARY_TYPE;
2683     SECONUM      : INTEGER
2684 ) : BOOLEAN;
2685
2686 VAR
2687     TEST          : LINE_OBJECT;
2688     FOUND_IT      : BOOLEAN;
2689     PAST_IT       : BOOLEAN;
2690
2691 BEGIN
2692     { +
2693     Stuff test object for comparison routine.
2694     - }
2695     TEST.OBJECT_TYPE := OBJ_TYP;
2696     TEST.PRIMARY     := PRIM;
2697     TEST.PRINUM      := PRIMNUM;
2698     TEST.SECONDARY   := SECO;
2699     TEST.SECNUM      := SECONUM;
2700
2701     { +
2702     Start looking at head of list.
2703     - }
2704     DEF_CURRENT      := DEF_HEAD;
2705     FOUND_IT         := FALSE;
2706     PAST_IT          := FALSE;
2707
2708     IF DEF_CURRENT <> NIL THEN
2709     BEGIN
2710         REPEAT
2711             FOUND_IT := CURRENT_EQ_TEST (TEST,TRUE);
2712             PAST_IT  := CURRENT_GT_TEST (TEST,TRUE);
2713
2714             IF NOT FOUND_IT THEN
2715                 INCR_CURRENT;
2716
2717             UNTIL (FOUND_IT) OR (PAST_IT) OR (DEF_CURRENT = NIL);
2718
2719         END;
2720
2721     { +
2722     Function value indicates whether we found it or not.
2723     - }
2724     FIND_OBJECT      := FOUND_IT;
2725
2726 END; { FIND_OBJECT }
```

```
2733 { ++
2734
2735 POINT_AT_DEFINITION -- Setup the list pointers.
2736
2737 This routine makes the list pointers point at the Definition Linked List.
2738
2739 CALLING SEQUENCE:
2740
2741 POINT_AT_DEFINITION;
2742
2743 INPUT PARAMETERS:
2744
2745 none
2746
2747 IMPLICIT INPUTS:
2748
2749 none
2750
2751 OUTPUT PARAMETERS:
2752
2753 none
2754
2755 IMPLICIT OUTPUTS:
2756
2757 DEF_HEAD
2758 DEF_TAIL
2759 DEF_ANL_HEAD
2760 DEF_ANL_TAIL
2761 POINTING_DIRECTION
2762
2763 ROUTINES CALLED:
2764
2765 none
2766
2767 ROUTINE VALUE:
2768
2769 none
2770
2771 SIGNALS:
2772
2773 none
2774
2775 SIDE EFFECTS:
2776
2777 the current list is the definition list
2778
2779 -- }
```


EDFUTIL
V04-000

Source Listing

J 3
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (54) Page 67

```
2781 [GLOBAL] PROCEDURE POINT_AT_DEFINITION;  
2782  
2783 BEGIN  
2784     IF NOT POINTING_AT_DEFINITION THEN  
2785     BEGIN  
2786         DEF_ANL_HEAD      := DEF_HEAD;  
2787         DEF_ANL_TAIL      := DEF_TAIL;  
2788         DEF_HEAD          := DEF_SAVE_HEAD;  
2789         DEF_TAIL          := DEF_SAVE_TAIL;  
2790  
2791         POINTING_AT_DEFINITION := TRUE;  
2792  
2793     END;  
2794  
2795 END; { POINT_AT_DEFINITION }  
2796  
2797  
2798
```

```
2800 { ++
2801
2802 POINT_AT_ANALYSIS -- Setup the list pointers.
2803
2804 This routine makes the list pointers point at the Analysis Linked List.
2805
2806 CALLING SEQUENCE:
2807
2808 POINT_AT_ANALYSIS;
2809
2810 INPUT PARAMETERS:
2811
2812 none
2813
2814 IMPLICIT INPUTS:
2815
2816 none
2817
2818 OUTPUT PARAMETERS:
2819
2820 none
2821
2822 IMPLICIT OUTPUTS:
2823
2824 DEF_HEAD
2825 DEF_TAIL
2826 DEF_ANL_HEAD
2827 DEF_ANL_TAIL
2828 POINTING_DIRECTION
2829
2830 ROUTINES CALLED:
2831
2832 none
2833
2834 ROUTINE VALUE:
2835
2836 none
2837
2838 SIGNALS:
2839
2840 none
2841
2842 SIDE EFFECTS:
2843
2844 the current list is the analysis list
2845
2846 -- }
```


EDFUTIL
V04-000

Source Listing

L 3
16-Sep-1984 00:51:37
5-Sep-1984 13:38:55

VAX-11 Pascal V2.4-277
DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (56) Page 69

2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865

```
PROCEDURE POINT_AT_ANALYSIS;  
BEGIN  
    IF POINTING_AT_DEFINITION THEN  
        BEGIN  
            DEF_SAVE_HEAD      := DEF_HEAD;  
            DEF_SAVE_TAIL      := DEF_TAIL;  
            DEF_HEAD           := DEF_ANL_HEAD;  
            DEF_TAIL           := DEF_ANL_TAIL;  
            POINTING_AT_DEFINITION := FALSE;  
        END;  
    END;  
END; { POINT_AT_ANALYSIS }
```

```
2867 { ++
2868
2869 EDF$LINE_PARSED -- Action routine for FDL$PARSE routine.
2870
2871 This routine stores into the definition database the values from FDL$PARSE.
2872
2873 CALLING SEQUENCE:
2874
2875 Called from the FDL$PARSE routine.
2876
2877 INPUT PARAMETERS:
2878
2879 none
2880
2881 IMPLICIT INPUTS:
2882
2883 FDL_BLOCK
2884
2885 OUTPUT PARAMETERS:
2886
2887 none
2888
2889 IMPLICIT OUTPUTS:
2890
2891 DEF_CURRENT
2892 DEF_TAIL
2893 DEF_HEAD
2894 DEF_SCRATCH
2895
2896 ROUTINES CALLED:
2897
2898 MAKE_SCRATCH
2899 LIB$COPY_DXDX
2900
2901 ROUTINE VALUE:
2902
2903 Always 1 (for success).
2904
2905 SIGNALS:
2906
2907 none
2908
2909 SIDE EFFECTS:
2910
2911 none
2912
2913 -- }
```



```
[ASYNCHRONOUS,GLOBAL] FUNCTION EDF$LINE_PARSED : INTEGER;
BEGIN
    { +
    This routine always succeeds.
    - }
    EDF$LINE_PARSED := 1;

    { +
    Create a new line object to be added to the list.
    - }
    MAKE_SCRATCH;

    { +
    Get the control longword.
    - }
    TEMP_FDL3$TYPE := FDL_BLOCK^[FDL$SL_CTRL]::FDL3$TYPE;

    { +
    Completely ignore a line if it's an IDENT, or if the Warning bit is set.
    - }
    IF ( NOT (
        ((TEMP_FDL3$TYPE.FDL$V_NEWPRI)
        AND
        (FDL_BLOCK^[FDL$SL_PRIMARY]::PRIMARY_TYPE = IDENT))
        OR
        (TEMP_FDL3$TYPE.FDL$V_WARNING)
    )) THEN
    WITH DEF_SCRATCH^ DO
    BEGIN
        { +
        Set the type of this line_object.
        - }
        IF TEMP_FDL3$TYPE.FDL$V_NEWPRI THEN
            OBJECT_TYPE := PRI
        ELSE
            OBJECT_TYPE := SEC;

        { +
        Check for a full line comment, as it is a 3rd object type.
        - }
        IF TEMP_FDL3$TYPE.FDL$V_LINECMT THEN
            OBJECT_TYPE := COM;

        { +
        Fetch the primary and secondary values.
        - }
        PRIMARY := FDL_BLOCK^[FDL$SL_PRIMARY]::PRIMARY_TYPE;
        SECONDARY := FDL_BLOCK^[FDL$SL_SECONDARY]::SECONDARY_TYPE;
```

```
2972 { +
2973 Store the comment string if a comment was detected.
2974 - }
2975 IF TEMP_FDL3$TYPE.FDL$V_COMMENT OR TEMP_FDL3$TYPE.FDL$V_LINECMT THEN
2976 BEGIN
2977     TEMP_DESCRIPTOR      := NULL_STRING;
2978     TEMP_DESCRIPTOR      := CVT_QUAD_DESC (
2979                                     FDL_BLOCK^[FDL$Q_COMMENT],
2980                                     FDL_BLOCK^[FDL$Q_COMMENT+1]
2981                                     );
2982     LIB$SCOPY_DXDX (TEMP_DESCRIPTOR,COMMENT);
2983
2984 END;
2985 { +
2986 Store the string if it's an attribute with a string value.
2987 - }
2988 IF (
2989     ((NOT TEMP_FDL3$TYPE.FDL$V_NEWPRI) AND (SEC_TYPE[SECONDARY].STR))
2990 OR
2991     { for positioning by file name }
2992     ((NOT TEMP_FDL3$TYPE.FDL$V_NEWPRI) AND (SECONDARY = POSITION$))
2993 OR
2994     ((TEMP_FDL3$TYPE.FDL$V_NEWPRI) AND (PRIMARY = TITLE))
2995 ) THEN
2996 BEGIN
2997     TEMP_DESCRIPTOR      := NULL_STRING;
2998     TEMP_DESCRIPTOR      := CVT_QUAD_DESC (
2999                                     FDL_BLOCK^[FDL$Q_STRING],
3000                                     FDL_BLOCK^[FDL$Q_STRING+1]
3001                                     );
3002     LIB$SCOPY_DXDX (TEMP_DESCRIPTOR,STRING);
3003
3004 END;
3005 { +
3006 Now stuff the new line_object with the remaining data from FDL$PARSE.
3007 - }
3008 IF PRIMARY IN [ AREA, KEY, ANALYSIS_OF_AREA, ANALYSIS_OF_KEY ] THEN
3009     PRINUM      := FDL_BLOCK^[FDL$L_PRINUM]
3010 ELSE
3011     PRINUM      := 0;
3012 IF SECONDARY IN [ SEG_LENGTH, SEG_POSITION ] THEN
```



```
3029
3030     SECNUM      := FDL_BLOCK^[FDL$SECNUM]
3031
3032 { +
3033 Until RMS supports different types per segment,
3034 make the Type secondary that last one in the Key primary.
3035 (SECNUM is higher sorting priority than SECONDARY)
3036 - }
3037 ELSE IF SECONDARY = SEG_TYPE THEN
3038
3039     SECNUM      := 7
3040
3041 ELSE
3042
3043     SECNUM      := 0;
3044
3045 { ++
3046
3047 IF RMS EVER IMPLEMENTS DIFFERENT DATA TYPES FOR EACH KEY SEGMENT,
3048 USE THE FOLLOWING CODE.
3049
3050     IF SECONDARY IN [ SEG_LENGTH, SEG_POSITION, SEG_TYPE ] THEN
3051
3052         SECNUM      := FDL_BLOCK^[FDL$SECNUM]
3053
3054     ELSE
3055
3056         SECNUM      := 0;
3057
3058 -- }
```

```
3060 { +
3061 Qualifiers values are more complicated.
3062 - }
3063 IF (
3064 (SECONDARY = MT_PROTECTION)
3065 OR
3066 (SECONDARY = NULL_VALUE)
3067 ) THEN
3068 BEGIN
3069     { +
3070     These two come back in a funny place.
3071     - }
3072     QUALIFIER := 0;
3073     NUMBER := FDL_BLOCK^[FDL$SL_QUALIFIER];
3074 END
3075 ELSE
3076 BEGIN
3077     { +
3078     See which secondary we have.
3079     - }
3080     CASE SECONDARY OF
3081     ORGANIZATION :
3082         CASE FDL_BLOCK^[FDL$SL_QUALIFIER] OF
3083             FAB$C_IDX : QUALIFIER := FDL$C_IDX;
3084             FAB$C_REL : QUALIFIER := FDL$C_REL;
3085             FAB$C_SEQ : QUALIFIER := FDL$C_SEQ;
3086         OTHERWISE
3087             { NULL-STATEMENT } ;
3088         END; { CASE FDL_BLOCK^[FDL$SL_QUALIFIER] }
3089     FORMAT :
3090         CASE FDL_BLOCK^[FDL$SL_QUALIFIER] OF
3091             FAB$C_FIX : QUALIFIER := FDL$C_FIX;
3092             FAB$C_STM : QUALIFIER := FDL$C_STM;
3093             FAB$C_STMCR : QUALIFIER := FDL$C_STMCR;
3094             FAB$C_STMLF : QUALIFIER := FDL$C_STMLF;
3095             FAB$C_UDF : QUALIFIER := FDL$C_UDF;
3096             FAB$C_VAR : QUALIFIER := FDL$C_VAR;
3097             FAB$C_VFC : QUALIFIER := FDL$C_VFC;
3098         OTHERWISE
3099             { NULL-STATEMENT } ;
```



```
3117      END;          { CASE FDL_BLOCK^[FDL$$_QUALIFIER] }
3118
3119      SEG_TYPE :
3120
3121      CASE FDL_BLOCK^[FDL$$_QUALIFIER] OF
3122
3123          XAB$C_BN2 :      QUALIFIER      := FDL$C_BN2;
3124          XAB$C_BN4 :      QUALIFIER      := FDL$C_BN4;
3125          XAB$C_BN8 :      QUALIFIER      := FDL$C_BN8;
3126          XAB$C_PAC :      QUALIFIER      := FDL$C_PAC;
3127          XAB$C_IN2 :      QUALIFIER      := FDL$C_IN2;
3128          XAB$C_IN4 :      QUALIFIER      := FDL$C_IN4;
3129          XAB$C_IN8 :      QUALIFIER      := FDL$C_IN8;
3130          XAB$C_STG :      QUALIFIER      := FDL$C_STG;
3131
3132      OTHERWISE
3133
3134          { NULL-STATEMENT } ;
3135
3136      END;          { CASE FDL_BLOCK^[FDL$$_QUALIFIER] }
3137
3138      OTHERWISE
3139
3140          QUALIFIER      := FDL_BLOCK^[FDL$$_QUALIFIER];
3141
3142      END;          { CASE SECONDARY }
3143
3144      NUMBER      := FDL_BLOCK^[FDL$$_NUMBER];
3145
3146      END;
3147
3148      { +
3149      Now store the other information coming back from FDL$PARSE.
3150      - }
3151      IF ODD (FDL_BLOCK^[FDL$$_SWITCH]) THEN
3152
3153          SWITCH      := TRUE
3154
3155      ELSE
3156
3157          SWITCH      := FALSE;
3158
3159      OWNER_UIC      := FDL_BLOCK^[FDL$$_OWNER_UIC];
3160      PROT_MASK      := FDL_BLOCK^[FDL$$_PROTECTION]::CTRL_ARRAY;
3161      FID1           := FDL_BLOCK^[FDL$$_FID1];
3162      FID2           := FDL_BLOCK^[FDL$$_FID2];
3163      FID3           := FDL_BLOCK^[FDL$$_FID3];
3164
3165      { +
3166      Now put def_scratch into the linked list. Depending upon whether
3167      we're inputting an FDL file or a analysis file.
3168      CLUSTER_SIZE must go into both the analysis definition and the
3169      main definition.
3170      - }
3171      IF (
3172      (ANALYSIS_ONLY AND (PRIMARY IN [ ANALYSIS_OF_KEY, ANALYSIS_OF_AREA ]))
```

```
3174 OR  
3175 ((NOT ANALYSIS_ONLY) AND (NOT (PRIMARY IN [ ANALYSIS_OF_KEY,  
3176 ANALYSIS_OF_AREA ])))  
3177 OR  
3178 (SECONDARY IN [ CLUSTER_SIZE ] ) THEN  
3179  
3180 IF OBJECT_TYPE = COM THEN  
3181  
3182 BEGIN  
3183  
3184 DEF_CURRENT := DEF_TAIL;  
3185 INSERT_AFTER_CURRENT;  
3186  
3187 END  
3188  
3189 ELSE  
3190  
3191 INSERT_IN_ORDER (IGNORE_OBJ);  
3192  
3193 END;  
3194 { IF NOT IDENT PRIMARY; ALSO WITH DEF_SCRATCH^ DO }  
3195  
3196 END; { EDF$LINE_PARSED }  
3197  
3198 END.  
3199 { End of file: SRC$:EDFUTIL.PAS }
```



```
.TITLE EDFUTIL
.IDENT \V04-000\

00000 .PSECT $CODE,PIC,CON,REL,LCL,SHR,EXE,RD,NOWRT,2

00000 C.AAA: .ASCII <27><92>
00002 C.AAB: .ASCII \
00004 C.AAC: .ASCII \ Lines\<0><0>
0000C C.AAD: .ASCII <27>\Pp;S(E);\<27><92><0>
00018 C.AAE: .ASCII <27>\PpP[27,320];V(W(10,S1,E,S[,479]))[+7\-\
00026 \67];\<27><92><0>
00034
00042
00044 C.AAF: .ASCII <27>\Pp;S(E);\<27><92><0>
00050 C.AAG: .ASCII \ There is no Primary Key in the Current \-
0005E \Definition. \
0006C
0007A
00084 C.AAH: .BYTE ^X38,8
00086 .BLKB 2
00088 C.AAI: .LONG 0,0,0,0
00098 .BYTE ^X60,0,0
0009B .BLKB 1
0009C C.AAJ: .LONG 0,0,^X800,0
000AC .BYTE 0,0,0
000AF .BLKB 1
000B0 C.AAK: .BYTE ^X18,0
000B2 C.AAL: .BYTE ^X18,0

000C 00000 NUM_LEN: .WORD ^M<R2,R3> ; 0166
5C 04 BC D0 00002 MOVL @4(R12),NUMBER
00V 12 00006 BNEQ 2$ ; 0176
50 01 D0 00008 MOVL #1,NUM_LEN ; 0181
00V 11 0000B BRB 10$
51 3B9ACA00 8F D0 0000D 2$: MOVL #1000000000,TEST_VAR ; 0190
52 0A D0 00014 MOVL #10,TEST_LEN ; 0191
53 5C D0 00017 MOVL NUMBER,R3 ; 0193
00V 18 0001A BGEQ 4$
53 53 CE 0001C MNEGL R3,R3
51 53 D1 0001F 4$: CMPL R3,TEST_VAR ; 0195
00V 18 00022 BGEQ 6$
52 D7 00024 DECL TEST_LEN ; 0197
51 0A C6 00026 6$: DIVL2 #10,TEST_VAR ; 0199
51 53 D1 00029 CMPL R3,TEST_VAR
F1 19 0002C BLSS 4$
5C D5 0002E TSTL NUMBER ; 0206
00V 18 00030 BGEQ 9$
52 D6 00032 INCL TEST_LEN ; 0208
50 52 D0 00034 9$: MOVL TEST_LEN,NUM_LEN ; 0213
04 00037 10$: RET ; 0217

00000 MAX_FACTOR: ; 0267
001C 00000 .WORD ^M<R2,R3,R4>
50 04 BC D0 00002 MOVL @4(R12),BASE
51 08 BC D0 00006 MOVL @8(R12),VALUE
```


	5C	0C	BC	D0	0000A	MOVL	@12(R12),MAX		
	50		51	D1	0000E	CMPL	VALUE,BASE		: 0278
			00V	19	00011	BLSS	2\$		
			50	D5	00013	TSTL	BASE		
			00V	12	00015	BNEQ	3\$		
	52		50	D0	00017	MOVL	BASE,TEMP		: 0280
			00V	11	0001A	BRB	7\$		
	51	52	50	C7	0001C	DIVL3	BASE,VALUE,TEMP		: 0286
53	51	51	00	7A	00020	EMUL	#0,#0,VALUE,R3		: 0288
53	53	53	50	7B	00025	EDIV	BASE,R3,R3,R3		
			53	D5	0002A	TSTL	R3		
			00V	18	0002C	BGEQ	4\$		
	53		50	C0	0002E	ADDL2	BASE,R3		
			53	D5	00031	TSTL	R3		
			00V	13	00033	BEQL	6\$		
			52	D6	00035	INCL	TEMP		: 0290
	52		50	C4	00037	MULL2	BASE,TEMP		: 0292
	5C		52	D1	0003A	CMPL	TEMP,MAX		: 0296
			00V	15	0003D	BLEQ	9\$		
	52		5C	D0	0003F	MOVL	MAX,TEMP		: 0298
	50		52	D0	00042	MOVL	MAX_FACTOR,R0		: 0302
			04	00045	RET				

; Routine Size: 70 bytes, Routine Base: \$CODE + 000EC

				00000	CALC_REC_OVERHEAD:				: 0348
	5C	04	BC	D0	00002	WORD	^M<>		
			50	D4	00006	MOVL	@4(R12),INDEX_LEVEL		
		00000084G	EF	D5	00008	CLRL	RECORD_OVERHEAD		: 0357
			00V	13	0000E	TSTL	IDATA+T32		: 0362
			5C	D5	00010	BEQL	3\$		
			00V	12	00012	TSTL	INDEX_LEVEL		
			50	0B	00014	BNEQ	3\$		
00V00000000CG	EF		00	C0	00017	ADDL2	#11,RECORD_OVERHEAD		: 0364
			5C	D5	0001F	BBC	#0,BDATA+12,5\$: 0369
			00V	13	00021	TSTL	INDEX_LEVEL		
00V00000000EG	EF		00	E1	00023	BEQL	7\$		
			5C	D5	0002B	BBC	#0,BDATA+14,8\$		
			00V	13	0002D	TSTL	INDEX_LEVEL		
	50		02	C0	0002F	BEQL	8\$		
			5C	D5	00032	ADDL2	#2,RECORD_OVERHEAD		: 0375
			00V	13	00034	TSTL	INDEX_LEVEL		: 0380
	50		04	C0	00036	BEQL	10\$		
		00000084G	EF	D5	00039	ADDL2	#4,RECORD_OVERHEAD		: 0382
			00V	12	0003F	TSTL	IDATA+132		: 0387
			5C	D5	00041	BNEQ	21\$		
			00V	12	00043	TSTL	INDEX_LEVEL		
00V000000000G	EF		00	E1	00045	BNEQ	21\$		
	50		0B	C0	0004D	BBC	#0,VARIABLE_RECORDS,14\$: 0391
			00V	11	00050	ADDL2	#11,RECORD_OVERHEAD		: 0393
	50		09	C0	00052	BRB	15\$		
00V00000000DG	EF		00	E1	00055	ADDL2	#9,RECORD_OVERHEAD		: 0397
	50		03	C0	0005D	BBC	#0,BDATA+T3,17\$: 0399
00V00000000CG	EF		00	E0	00060	ADDL2	#3,RECORD_OVERHEAD		: 0401
00V00000000DG	EF		00	E1	00068	BBS	#0,BDATA+T2,19\$: 0403
	50	000000D8G	EF	C0	00070	BBC	#0,BDATA+13,21\$		
						ADDL2	IDATA+216,RECORD_OVERHEAD		: 0405


```
04 00077 21$: RET ; 0411

; Routine Size: 120 bytes, Routine Base: $CODE + 00132

00000 CALC_BUC_OVERHEAD: ; 0457
0000 .WORD ^M<>
5C 04 BC D0 00002 MOVL @4(R12),INDEX_LEVEL ; 0463
00V 12 00006 BNEQ 2$ ; 0465
5C 10 D0 00008 MOVL #16,CALC_BUC_OVERHEAD ; 0469
00V 11 0000B BRB 3$ ; 0471
5C 12 D0 0000D 2$: MOVL #18,CALC_BUC_OVERHEAD
50 5C D0 00010 3$: MOVL CALC_BUC_OVERHEAD,R0
04 00013 RET

; Routine Size: 20 bytes, Routine Base: $CODE + 001AA

0000 00000 .ENTRY EDF$RESET_SCROLL,^M<> ; 0520
00V00000000G SE 08 C2 00002 SUBL2 #8,SP
EF 00 E0 00005 BBS #0,AUTO_TUNE,8$ ; 0524
00V00000000G EF 00 E1 0000D BBC #0,REGIS,4$ ; 0531
00000000G EF D4 00015 CLRL CHFFLAGS ; 0535
00000000G EF B4 0001B CLRW OUT_LINE ; 0536
FFFFFFE1B EF 9F 00021 PUSHAB C,AAA
02 DD 00027 PUSHL #2
00000000G EF 9F 00029 PUSHAB OUT_LINE
000000FF 8F DD 0002F PUSHL #255
00000000G EF 04 FB 00035 CALLS #4,PASS$WRITEV_STRING ; 0537
00000000G EF 9F 0003C PUSHAB CHFFLAGS
00000000G EF 9F 00042 PUSHAB ONE
F8 AD 0B2500FF 8F D0 00048 MOVL #186974463,-8(FP)
FC AD 00000000G EF 9E 00050 MOVAB OUT_LINE,-4(FP)
F8 AD 9F 00058 PUSHAB -8(FP)
00000000G EF 03 FB 0005B CALLS #3,LIB$PUT_LINE
00V00000000G EF 00 E1 00062 4$: BBC #0,SCROLLING_SET,7$ ; 0544
00000000G EF 9F 0006A PUSHAB LINES_PER_PAGE ; 0546
00000000G EF 9F 00070 PUSHAB LINE_ONE
00000000G EF 02 FB 00076 CALLS #2,LIB$SET_SCROLL
03 00000000G EF 00 E0 0007D 7$: BBS #0,FILE_CREATED,+.3 ; 0553
0000V 31 00085 8$: BRW 12$
00000000G EF B5 00088 TSTW RES_OUTPUT_FILENAME_DESC
03 1A 0008E BGTRU .+3
0000V 31 00090 BRW 12$
00000000G EF 02 D0 00093 MOVL #2,CHFFLAGS ; 0561
00000000G EF B4 0009A CLRW OUT_LINE ; 0562
00000000G EF 9F 000A0 PUSHAB CRLF
02 DD 000A6 PUSHL #2
00000000G EF 9F 000A8 PUSHAB OUT_LINE
000000FF 8F DD 000AE PUSHL #255
00000000G EF 04 FB 000B4 CALLS #4,PASS$WRITEV_STRING
7E 00000000G EF 3C 000BB MOVZWL RES_OUTPUT_FILENAME_DESC,-(SP)
00 DD 000C2 PUSHL #0
50 00000004G EF D0 000C4 MOVL RES_OUTPUT_FILENAME_DESC+4,R0
60 9F 000CB PUSHAB (R0)
000000FF 8F DD 000CD PUSHL #255
00000000G EF 9F 000D3 PUSHAB OUT_LINE
000000FF 8F DD 000D9 PUSHL #255
```

```
Generated Code
00000000G EF FFFFFFFD58 06 FB 000DF CALLS #6,PASSWRITEV_STRING
EF 9F 000E6 PUSHAB C.AAB
02 DD 000EC PUSHL #2
EF 9F 000EE PUSHAB OUT_LINE
8F DD 000F4 PUSHL #255
00000000G EF 04 FB 000FA CALLS #4,PASSWRITEV_STRING
00000000G EF 9F 00101 PUSHAB LINES_SHOWN
00B4 CF 01 FB 00107 CALLS #1,NUM_LEN
50 DD 0010C PUSHL R0
00000000G EF DD 0010E PUSHL LINES_SHOWN
00000000G EF 9F 00114 PUSHAB OUT_LINE
00000000G 8F DD 0011A PUSHL #255
00000000G EF 04 FB 00120 CALLS #4,PASSWRITEV_INTEGER
FFFFFFD19 EF 9F 00127 PUSHAB C.AAC
06 DD 0012D PUSHL #6
00000000G EF 9F 0012F PUSHAB OUT_LINE
00000000G 8F DD 00135 PUSHL #255
00000000G EF 04 FB 0013B CALLS #4,PASSWRITEV_STRING
00000000G EF 9F 00142 PUSHAB CHFFLAGS
00000000G EF 9F 00148 PUSHAB ONE
F8 AD 0B2500FF 8F D0 0014E MOVL #186974463,-8(FP)
FC AD 00000000G EF 9E 00156 MOVAB OUT_LINE,-4(FP)
F8 AD 9F 0015E PUSHAB -8(FP)
00000000G EF 03 FB 00161 CALLS #3,LIB$PUT_LINE
04 00168 12$: RET ; 0565
; 0569
```

; Routine Size: 361 bytes, Routine Base: \$CODE + 001BE

```
0000 00000 CLEAR: .WORD ^M<> ; 0617
03 00000000G SC 04 BC D0 00002 MOVL @4(R12),DESTINATION ; 0626
EF 00 E0 00006 BBS #0,VIDEO_TERMINAL,..+3
0000V 31 0000E BRW 28$
03 00000000G EF 00 E1 00011 BBC #0,AUTO_TUNE,..+3
0000V 31 00019 BRW 28$
03 00 SC CF 0001C CASEL DESTINATION,#0,#3 ; 0634
0000V 00020 .DISPL 14$
0000V 00022 .DISPL 8$
0000V 00024 .DISPL 24$
0000V 00026 .DISPL 3$
0000V 31 00028 BRW 26$
00V00000000G EF 00 E1 0002B 3$: BBC #0,REGIS,5$ ; 0646
FFFFFFCAC EF 9F 00033 ; 0648
0B DD 00039 PUSHAB C.AAD
EF 9F 0003B PUSHL #11
00000000G EF 03 FB 00041 PUSHAB PASSFV OUTPUT
00000000G EF 9F 00048 CALLS #3,PASSWRITE STRING
00000000G EF 01 FB 0004E CALLS #1,PASSWRITELN2
00000000G EF 9F 00055 5$: PUSHAB COL ONE ; 0650
00000000G EF 9F 0005B PUSHAB LINE ONE
00000000G EF 02 FB 00061 CALLS #2,LIB$ERASE_PAGE ; 0651
01 DD 00068 PUSHL #1
20 DD 0006A PUSHL #32
00000000G EF 9F 0006C PUSHAB PASSFV OUTPUT
00000000G EF 03 FB 00072 CALLS #3,PASSWRITE CHAR
00000000G EF 9F 00079 PUSHAB PASSFV OUTPUT
00000000G EF 01 FB 0007F CALLS #1,PASSWRITELN2 ; 0652
00000000G EF 9F 00086 PUSHAB COL_ONE
```


00000000G	EF	00000000G	EF	9F 0008C	PUSHAB	LINE ONE	
			02	FB 00092	CALLS	#2,LIB\$SET_CURSOR	
00V00000000G	EF	0000V	31	00099	BRW	27\$	
			00	E1 0009C	BBC	#0,REGIS,11\$: 0660
		FFFFFFC47	EF	9F 000A4	PUSHAB	C.AAE	: 0664
			2B	DD 000AA	PUSHL	#43	
00000000G	EF	00000000G	EF	9F 000AC	PUSHAB	PASSFV OUTPUT	
			03	FB 000B2	CALLS	#3,PASSWRITE_STRING	
00000000G	EF	00000000G	EF	9F 000B9	PUSHAB	PASSFV OUTPUT	
			01	FB 000BF	CALLS	#1,PASSWriteln2	
00000000G	EF	00000000G	EF	9F 000C6	PUSHAB	COL ONE	: 0667
		00000000G	EF	9F 000CC	PUSHAB	PROMPT LINE	
00000000G	EF	0000V	02	FB 000D2	CALLS	#2,LIB\$SET_CURSOR	
			31	000D9	BRW	27\$	
		00000000G	EF	9F 000DC	PUSHAB	COL ONE	: 0673
		00000000G	EF	9F 000E2	PUSHAB	LOWER LINE	
00000000G	EF	0000V	02	FB 000E8	CALLS	#2,LIB\$ERASE_PAGE	
			31	000EF	BRW	27\$	
00V00000000G	EF	00	E0	000F2	BBS	#0,FULL_PROMPT,16\$: 0677
03 00000000G	EF	00	E0	000FA	BBS	#0,TEMP_FULL_PROMPT,..+3	
		0000V	31	00102	BRW	27\$	
00V00000000G	EF	00	E1	00105	BBC	#0,TEMP_FULL_PROMPT,18\$: 0681
		666640A6	8F	DF 0010D	PUSHAF	#^F1.3	: 0683
00000000G	EF	01	FB	00113	CALLS	#1,LIB\$WAIT	
00V00000000G	EF	00	E1	0011A	BBC	#0,REGIS,20\$: 0692
		FFFFFFBF5	EF	9F 00122	PUSHAB	C.AAF	: 0694
			0B	DD 00128	PUSHL	#11	
00000000G	EF	00000000G	EF	9F 0012A	PUSHAB	PASSFV OUTPUT	
			03	FB 00130	CALLS	#3,PASSWRITE_STRING	
00000000G	EF	00000000G	EF	9F 00137	PUSHAB	PASSFV OUTPUT	
			01	FB 0013D	CALLS	#1,PASSWriteln2	
00000000G	EF	00000000G	EF	9F 00144	PUSHAB	COL ONE	: 0696
		00000000G	EF	9F 0014A	PUSHAB	LINE ONE	
00000000G	EF	0000V	02	FB 00150	CALLS	#2,LIB\$ERASE_PAGE	
			01	DD 00157	PUSHL	#1	: 0697
			20	DD 00159	PUSHL	#32	
00000000G	EF	00000000G	EF	9F 0015B	PUSHAB	PASSFV OUTPUT	
			03	FB 00161	CALLS	#3,PASSWRITE_CHAR	
00000000G	EF	00000000G	EF	9F 00168	PUSHAB	PASSFV OUTPUT	
			01	FB 0016E	CALLS	#1,PASSWriteln2	
00000000G	EF	00000000G	EF	9F 00175	PUSHAB	COL ONE	: 0698
		00000000G	EF	9F 0017B	PUSHAB	LINE ONE	
00000000G	EF	0000V	02	FB 00181	CALLS	#2,LIB\$SET_CURSOR	
			11	00188	BRB	27\$	
00000000G	EF	0000001F	8F	DF 0018A	PUSHAL	#31	: 0702
			01	FB 00190	CALLS	#1,QUERY	
		00V	11	00197	BRB	27\$	
				00199		26\$:	
				00199		27\$:	
			04	00199	RET	28\$:	: 0712

; Routine Size: 410 bytes, Routine Base: \$CODE + 00327

			0000	0000	CVT_QUAD_DESC:	
			0000	0000	-WORD	^M<>
5E		08	C2	00002	SUBL2	#8,SP
50	04	BC	D0	00005	MOVL	@4(R12),LONG1

: 0759

	5C	08	BC	D0	00009	MOVL	@8(R12),LONG2	
		00000000G	EF	94	0000D	CLRB	QUAD_DESC	: 0770
00000001G	EF		50	D0	00013	MOVL	LONG1,QUAD_DESC+1	: 0771
00000005G	EF		5C	D0	0001A	MOVL	LONG2,QUAD_DESC+5	: 0772
00000000G	EF		01	90	00021	MOVB	#1,QUAD_DESC	: 0777
F8	AD	00000001G	EF	7D	00028	MOVQ	QUAD_DESC+1,CVT_QUAD_DESC	: 0778
	50	F8	AD	7D	00030	MOVQ	CVT_QUAD_DESC,R0	: 0782
				04	00034	RET		

; Routine Size: 53 bytes, Routine Base: \$CODE + 004C1

				00000	SCAN_DEFINITION:		: 0833	
				0000	00000	.WORD		
	5C	04	BC	90	00002	MOVB	@4(R12),FATAL	
00000000G	EF	00000000G	EF	D0	00006	MOVL	DEF_HEAD,DEF_CURRENT	: 0840
		00000000G	EF	94	00011	CLRB	FOUND_0	: 0841
		00000000G	EF	94	00017	CLRB	FOUND_AREA	: 0842
		00000000G	EF	94	0001D	CLRB	FOUND_KEY	: 0843
		00000000G	EF	D4	00023	CLRL	LOW_AREA	: 0844
		00000000G	EF	D4	00029	CLRL	HIGH_AREA	: 0845
		00000000G	EF	D4	0002F	CLRL	LOW_KEY	: 0846
		00000000G	EF	D4	00035	CLRL	HIGH_KEY	: 0847
	50	00000000G	EF	D0	0003B	1\$: MOVL	DEF_CURRENT,R0	: 0851
			60	95	00042	TSTB	(R0)	: 0855
			00V	12	00044	BNEQ	11\$	
	0B	19	A0	91	00046	CMPB	25(R0),#11	
			00V	12	0004A	BNEQ	11\$	
		1A	A0	D5	0004C	TSTL	26(R0)	: 0863
			00V	12	0004F	BNEQ	6\$	
00000000G	EF		01	90	00051	MOVB	#1,FOUND_0	: 0865
00000000G	EF		01	90	00058	6\$: MOVB	#1,FOUND_KEY	: 0867
00000000G	EF	1A	A0	D1	0005F	CMPL	26(R0),LOW_KEY	: 0869
			00V	18	00067	BGEQ	8\$	
00000000G	EF	1A	A0	D0	00069	MOVL	26(R0),LOW_KEY	: 0871
00000000G	EF	1A	A0	D1	00071	8\$: CMPL	26(R0),HIGH_KEY	: 0873
			00V	15	00079	BLEQ	11\$	
00000000G	EF	1A	A0	D0	0007B	MOVL	26(R0),HIGH_KEY	: 0875
			60	95	00083	11\$: TSTB	(R0)	: 0879
			00V	12	00085	BNEQ	18\$	
	05	19	A0	91	00087	CMPB	25(R0),#5	
			00V	12	0008B	BNEQ	18\$	
00000000G	EF		01	90	0008D	MOVB	#1,FOUND_AREA	: 0887
00000000G	EF	1A	A0	D1	00094	CMPL	26(R0),LOW_AREA	: 0889
			00V	18	0009C	BGEQ	15\$	
00000000G	EF	1A	A0	D0	0009E	MOVL	26(R0),LOW_AREA	: 0891
00000000G	EF	1A	A0	D1	000A6	15\$: CMPL	26(R0),HIGH_AREA	: 0893
			00V	15	000AE	BLEQ	18\$	
00000000G	EF	1A	A0	D0	000B0	MOVL	26(R0),HIGH_AREA	: 0895
	50	00000000G	EF	D0	000B8	18\$: MOVL	DEF_CURRENT,R0	: 0901
00000000G	EF	01	A0	D0	000BF	MOVL	1(R0),DEF_CURRENT	
			03	13	000C7	BEQL	+3	
			FF6F	31	000C9	BRW	1\$	
	00V		5C	E8	000CC	BLBS	FATAL,21\$: 0905
		00000000G	EF	D5	000CF	TSTL	HIGH_KEY	
			03	12	000D5	BNEQ	+3	
			0000V	31	000D7	BRW	26\$	
03 00000000G	EF		00	E1	000DA	21\$: BBC	#0,FOUND_0,..+3	

		0000V	31	000E2	BRW	26\$	
		00000000G	EF	9F 000E3	PUSHAB	SHIFT	: 0913
			04	DD 000EB	PUSHL	#4	
		00000000G	EF	9F 000ED	PUSHAB	PASSFV_OUTPUT	
00000000G	EF		03	FB 000F3	CALLS	#3,PASSWRITE_STRING	
		00000000G	EF	9F 000FA	PUSHAB	ANSI_REVERSE	
			04	DD 00100	PUSHL	#4	
		00000000G	EF	9F 00102	PUSHAB	PASSFV_OUTPUT	
00000000G	EF		03	FB 00108	CALLS	#3,PASSWRITE_STRING	
		FFFFFFA45	EF	9F 0010F	PUSHAB	C.AAG	
			34	DD 00115	PUSHL	#52	
		00000000G	EF	9F 00117	PUSHAB	PASSFV_OUTPUT	
00000000G	EF		03	FB 0011D	CALLS	#3,PASSWRITE_STRING	
		00000000G	EF	9F 00124	PUSHAB	ANSI_RESET	
			04	DD 0012A	PUSHL	#4	
		00000000G	EF	9F 0012C	PUSHAB	PASSFV_OUTPUT	
00000000G	EF		03	FB 00132	CALLS	#3,PASSWRITE_STRING	
		00000000G	EF	9F 00139	PUSHAB	PASSFV_OUTPUT	
00000000G	EF		01	FB 0013F	CALLS	#1,PASSWRITELN2	
00V00000000G	EF		00	E1 00146	BBC	#0,AUTO_TUNE,24\$: 0917
			00	DD 0014E	PUSHL	#0	: 0919
			00	DD 00150	PUSHL	#0	
			00	DD 00152	PUSHL	#0	
		00B3801C	8F	DD 00154	PUSHL	#11763740	
00000000G	EF		04	FB 0015A	CALLS	#4,LIB\$STOP	
			00V	11 00161	BRB	26\$	
		00004140	8F	DF 00163	PUSHAF	#^F3.0	: 0925
00000000G	EF		01	FB 00169	CALLS	#1,LIB\$WAIT	: 0926
			00	DD 00170	PUSHL	#0	
			00	DD 00172	PUSHL	#0	
			00	DD 00174	PUSHL	#0	
		00B3804B	8F	DD 00176	PUSHL	#11763787	
00000000G	EF		04	FB 0017C	CALLS	#4,LIB\$SIGNAL	: 0932
			04	00183	26\$: RET		

; Routine Size: 388 bytes, Routine Base: \$CODE + 004F6

				00000	PARSE_INPUT:		: 0979
			001C	00000	.WORD	^M<R2,R3,R4>	
		5E	08	C2 00002	SUBL2	#8,SP	
		52	04	BC D0 00005	MOVL	@4(R12),KEY_TABLE	
		53	08	BC D0 00009	MOVL	@8(R12),STATE_TABLE	
		54	0C	BC 90 0000D	MOVB	@12(R12),DEFAULT_OK	
		5C	10	BC D0 00011	MOVL	@16(R12),DEFAULT_VALUE	
00000000G	EF	00000000G	EF	7D 00015	MOVQ	NULL_STRING,INPUT_DESC	: 0991
00V00000000G	EF		00	E1 00020	BBC	#0,TAKE_DEFAULTS,4\$: 0996
		00000104G	EF	D5 00028	TSTL	IDATA+260	
			00V	12 0002E	BNEQ	4\$	
		00V	54	E9 00030	BLBC	DEFAULT_OK,4\$	
		1F 00000000G	EF	D1 00033	CMPL	QTAB_OFFSET,#31	
			00V	12 0003A	BNEQ	5\$	
00V00000000G	EF		00	E1 0003C	BBC	#0,AUTO_TUNE,8\$: 1012
03 00000000G	EF		00	E1 00044	BBC	#0,AUTO_TUNE,..+3	
			0000V	31 0004C	BRW	12\$	
		00000000G	EF	9F 0004F	PUSHAB	CRLF	: 1016
			02	DD 00055	PUSHL	#2	
		00000000G	EF	9F 00057	PUSHAB	PASSFV_OUTPUT	

Generated Code			
00000000G	EF	03	FB 0005D
00000000G	EF	01	9F 00064
00000000G	EF	8F	FB 0006A
00000000G	EF	01	DF 00071
00000000G	EF	01	FB 00077
00V00000000G	EF	0000V	31 0007E
00000000G	EF	30	E0 00081 8\$:
00000000G	EF	01	9F 00089
00V00000000G	EF	31	E0 00096 9\$:
00000000G	EF	01	9F 0009E
00000000G	EF	00	FB 000A4
		00	DD 000AB
		00	DD 000AD
		00	DD 000AF
00000000G	EF	8F	DD 000B1
00000000G	EF	04	FB 000B7
00000000G	EF	8F	DD 000BE 11\$:
00000000G	EF	03	9F 000C4
00000000G	EF	03	9F 000CA
00000000G	EF	01	FB 000D0
00000000G	EF	01	9F 000D7
00000000G	EF	01	FB 000DD
00000000G	EF	02	9F 000E4
00000000G	EF	02	DD 000EA
00000000G	EF	03	9F 000EC
00000000G	EF	03	FB 000F2
00000000G	EF	01	9F 000F9
00000000G	EF	01	FB 000FF
00000000G	AD	010E00FF	8F D0 00106
00000000G	AD	00000000G	EF 9E 0010E
00000000G	AD	00000000G	EF 9F 00116
00000000G	EF	02	9F 00119
00000000G	EF	02	FB 0011F
00000000G	EF	02	9F 00126
00000000G	EF	02	9F 0012C
00000000G	EF	02	FB 00132
00V00000000G	EF	00	E1 00139 12\$:
00000000G	EF	00V	B5 00141
00000000G	EF	00V	1B 00147
00000000G	7E	00000000G	EF 3C 00149
00000000G	00	00000000G	DD 00150
00000000G	50	00000004G	EF D0 00152
00000000G	000000FF	60	9F 00159
00000000G	00000000G	8F	DD 0015B
00000000G	EF	05	9F 00161
00000000G	EF	01	FB 00167
00000000G	EF	01	9F 0016E
00000000G	00V	11	0017B
00000000G	EF	01	9F 0017D 15\$:
00000000G	EF	01	FB 00183
00000000G	00000000G	0018A	16\$:
00000000G	00V	12	00190 17\$:
00000000G	00V	54	E9 00192
00000000G	EF	5C	D0 00195
			CALLS #3,PASSWRITE_STRING
			PUSHAB PASSFV_OUTPUT
			CALLS #1,PASSWRITELN2
			PUSHAF #^F0.7 ; 1017
			CALLS #1,LIB\$WAIT
			BRW 12\$
			BBS #48,PASSFV_INPUT,9\$; 1027
			PUSHAB PASSFV_INPUT
			CALLS #1,PASS\$LOOK_AHEAD
			BBS #49,PASSFV_INPUT,11\$
			PUSHAB PASSFV_INPUT ; 1031
			CALLS #1,PASS\$RESET2
			PUSHL #0 ; 1032
			PUSHL #0
			PUSHL #0
			PUSHL #11763787
			CALLS #4,LIB\$SIGNAL
			PUSHL #255 ; 1036
			PUSHAB PASSFV_INPUT
			PUSHAB INPUT_STRING
			CALLS #3,PASS\$READ_STRING
			PUSHAB PASSFV_INPUT
			CALLS #1,PASS\$READLN2
			PUSHAB CRLF ; 1037
			PUSHL #2
			PUSHAB PASSFV_OUTPUT
			CALLS #3,PASSWRITE_STRING
			PUSHAB PASSFV_OUTPUT
			CALLS #1,PASSWRITELN2
			MOVL #17694975,-8(FP) ; 1038
			MOVAB INPUT_STRING,-4(FP)
			PUSHAB -8(FP)
			PUSHAB INPUT_DESC
			CALLS #2,STR\$TRIM
			PUSHAB INPUT_DESC ; 1039
			PUSHAB INPUT_DESC
			CALLS #2,STR\$UPCASE
			BBC #0,JOURNAL_ENABLED,17\$; 1047
			TSTW INPUT_DESC ; 1049
			BLEQU 15\$
			MOVZWL INPUT_DESC,-(SP) ; 1051
			PUSHL #0
			MOVL INPUT_DESC+4,R0
			PUSHAB (R0)
			PUSHL #255
			PUSHAB JOURNAL_FILE
			CALLS #5,PASSWRITE_STRING
			PUSHAB JOURNAL_FILE
			CALLS #1,PASSWRITELN2
			BRB 16\$
			PUSHAB JOURNAL_FILE ; 1058
			CALLS #1,PASSWRITELN2
			TSTW INPUT_DESC ; 1063
			BNEQ 22\$
			BLBC DEFAULT_OK,20\$; 1065
			MOVL DEFAULT_VALUE,INPUT_VALUE ; 1067


```
0000V 31 0019C BRW 28$
00 DD 0019F 20$: PUSH  #0 ; 1073
00 DD 001A1 PUSH  #0
00 DD 001A3 PUSH  #0
00 DD 001A5 PUSH  #11763776
00 FB 001AB CALLS  #4,LIB$SIGNAL
00V 11 001B2 BRB 28$
00000000G EF 00B38040 EF DD 001B4 22$: MOVL INPUT_DESC+4,PARAM_BLOCK+12 ; 1084
00000000G EF 00000004G EF 3C 001BF MOVZWL INPUT_DESC,PARAM_BLOCK+8 ; 1085
00000000G EF 00000000G EF DD 001CA PUSH  KEY_TABLE ; 1087
00000000G EF 00000000G EF DD 001CC PUSH  STATE_TABLE
00000000G EF 00000000G EF 9F 001CE PUSHAB PARAM_BLOCK
00000000G EF 00000020G EF 03 FB 001D4 CALLS  #3,LIB$TPARSE
00000000G EF 0000001CG EF 50 DD 001DB MOVL R0,ISTATUS
00000000G EF 00000000G EF DD 001E2 MOVL PARAM_BLOCK+32,INPUT_VALUE ; 1093
00V00000000G EF 00000000G EF DD 001ED MOVL PARAM_BLOCK+28,INPUT_NUMBER ; 1094
00V00000000G EF 00000000G EF E8 001F8 BLBS ISTATUS,28$ ; 1099
00V00000000G EF 00000000G EF 00 E1 001FF BBC #0,PARAM_BLOCK+6,25$ ; 1103
00V00000000G EF 00000000G EF 00 DD 00207 PUSH  #0 ; 1105
00V00000000G EF 00000000G EF 00 DD 00209 PUSH  #0
00V00000000G EF 00000000G EF 00 DD 0020B PUSH  #0
00V00000000G EF 00B38028 EF 8F DD 0020D PUSH  #11763752
00V00000000G EF 00000000G EF 04 FB 00213 CALLS  #4,LIB$SIGNAL
00V00000000G EF 00000000G EF 00V 11 0021A BRB 28$
00V00000000G EF 00000000G EF 00 DD 0021C 25$: PUSH  #0 ; 1109
00V00000000G EF 00000000G EF 00 DD 0021E PUSH  #0
00V00000000G EF 00000000G EF 00 DD 00220 PUSH  #0
00V00000000G EF 00B38030 EF 8F DD 00222 PUSH  #11763760
00V00000000G EF 00000000G EF 04 FB 00228 CALLS  #4,LIB$SIGNAL
00V00000000G EF 00000000G EF 04 0022F 28$: RET ; 1115
```

; Routine Size: 560 bytes, Routine Base: \$CODE + 0067A

```
00000 NUMBER_INPUT: ; 1161
000C 00000 .WORD ^M<R2,R3>
5E 08 C2 00002 SUBL2 #8,SP
52 08 BC 90 00005 MOV B 08(R12),DEFAULT_OK
53 0C BC D0 00009 MOVL 012(R12),DEFAULT_VALUE
00000000G EF 00000000G EF 7D 0000D MOVQ NULL_STRING,INPUT_DESC ; 1172
00V00000000G EF 00000104G EF 00 E1 00018 BBC #0,TAKE_DEFAULTS,3$ ; 1177
00V00000000G EF 00000104G EF D5 00020 TSTL IDATA+260
00V00000000G EF 00000104G EF 12 00026 BNEQ 3$
00V00000000G EF 00000104G EF 52 E8 00028 BLBS DEFAULT_OK,4$
03 00000000G EF 00000104G EF 00 E1 0002B 3$: BBC #0,AUTO_TUNE,7$
03 00000000G EF 00000104G EF 00 E1 00033 4$: BBC #0,AUTO_TUNE,7$+3 ; 1191
00000000G EF 00000104G EF 31 0003B BRW 11$ ; 1195
00000000G EF 00000104G EF 9F 0003E PUSHAB CRLF
00000000G EF 00000104G EF 02 DD 00044 PUSH  #2
00000000G EF 00000104G EF 9F 00046 PUSHAB PASSFV_OUTPUT
00000000G EF 00000104G EF 03 FB 0004C CALLS  #3,PASSWRITE_STRING
00000000G EF 00000104G EF 9F 00053 PUSHAB PASSFV_OUTPUT
00000000G EF 00000104G EF 01 FB 00059 CALLS  #1,PASSWRITELN2
00000000G EF 00000104G EF 8F DF 00060 PUSHAF #^F0.7 ; 1196
00000000G EF 00000104G EF 01 FB 00066 CALLS  #1,LIB$WAIT
00V00000000G EF 00000104G EF 31 0006D BRW 11$
00V00000000G EF 00000104G EF 30 E0 00070 7$: BBS #48,PASSFV_INPUT,8$ ; 1206
00000000G EF 00000104G EF 9F 00078 PUSHAB PASSFV_INPUT
```


Generated Code

00000000G	EF	01	FB 0007E	CALLS	#1,PASS\$LOOK_AHEAD	
00V00000000G	EF	31	E0 00085	BBS	#49,PASS\$FV_INPUT,10\$	
				PUSHAB	PASS\$FV_INPUT	: 1210
00000000G	EF	01	9F 0008D	PUSHL	#0	: 1211
		00	FB 00093	CALLS	#1,PASS\$RESET2	
		00	DD 0009A	PUSHL	#0	
		00	DD 0009C	PUSHL	#0	
		00	DD 0009E	PUSHL	#0	
		8F	DD 000A0	PUSHL	#11763787	
00000000G	EF	04	FB 000A6	CALLS	#4,LIB\$SIGNAL	
		8F	DD 000AD	PUSHL	#255	: 1215
		0000000FF	9F 000B3	PUSHAB	PASS\$FV_INPUT	
		00000000G	9F 000B9	PUSHAB	INPUT_STRING	
00000000G	EF	03	FB 000BF	CALLS	#3,PASS\$READ_STRING	
		EF	9F 000C6	PUSHAB	PASS\$FV_INPUT	
00000000G	EF	01	FB 000CC	CALLS	#1,PASS\$READLN2	
		00000000G	9F 000D3	PUSHAB	CRLF	: 1216
		02	DD 000D9	PUSHL	#2	
		00000000G	9F 000DB	PUSHAB	PASS\$FV_OUTPUT	
00000000G	EF	03	FB 000E1	CALLS	#3,PASS\$WRITE_STRING	
		00000000G	9F 000E8	PUSHAB	PASS\$FV_OUTPUT	
00000000G	EF	01	FB 000EE	CALLS	#1,PASS\$WRITELN2	
F8	AD	8F	D0 000F5	MOVL	#17694975,-8(FP)	: 1217
FC	AD	EF	9E 000FD	MOVAB	INPUT_STRING,-4(FP)	
		F8	9F 00105	PUSHAB	-8(FP)	
00000000G	EF	02	FB 0010E	PUSHAB	INPUT_DESC	
		00000000G	9F 00115	CALLS	#2,STR\$TRIM	
		00000000G	9F 0011B	PUSHAB	INPUT_DESC	: 1218
		02	FB 00121	PUSHAB	INPUT_DESC	
00000000G	EF	00	E1 0013E	CALLS	#2,STR\$UPCASE	
00000014G	EF	00	D0 00128	MOVL	INPUT_DESC+4,PARAM_BLOCK+20	: 1219
00000010G	EF	00	3C 00133	MOVZWL	INPUT_DESC,PARAM_BLOCK+16	: 1220
00V00000000G	EF	00	B5 00146	BBC	#0,JOURNAL_ENABLED,16\$: 1228
		00V	1B 0014C	TSTW	INPUT_DESC	: 1230
		7E	3C 0014E	BLEQU	14\$	
		00	DD 00155	MOVZWL	INPUT_DESC,-(SP)	: 1232
		50	D0 00157	PUSHL	#0	
		60	9F 0015E	MOVL	INPUT_DESC+4,R0	
		000000FF	DD 00160	PUSHAB	(R0)	
00000000G	EF	05	FB 0016C	PUSHL	#255	
		00000000G	9F 00166	PUSHAB	JOURNAL_FILE	
		00000000G	9F 00173	CALLS	#5,PASS\$WRITE_STRING	
00000000G	EF	01	FB 00179	PUSHAB	JOURNAL_FILE	
		00V	11 00180	CALLS	#1,PASS\$WRITELN2	
		00000000G	9F 00182	BRB	16\$	
00000000G	EF	01	FB 00188	PUSHAB	JOURNAL_FILE	: 1239
		00000000G	B5 0018F	CALLS	#1,PASS\$WRITELN2	
		00V	12 00195	TSTW	INPUT_DESC	: 1244
		52	E9 00197	BNEQ	21\$	
04	00V	53	D0 0019A	BLBC	DEFAULT_OK,19\$: 1246
BC		00V	11 0019E	MOVL	DEFAULT_VALUE,@4(R12)	: 1248
		00	DD 001A0	BRB	20\$	
		00	DD 001A2	PUSHL	#0	: 1254
		00	DD 001A4	PUSHL	#0	
00000000G	EF	8F	DD 001A6	PUSHL	#11763776	
		04	FB 001AC	CALLS	#4,LIB\$SIGNAL	
		00V	11 001B3	BRB	24\$	


```

                                04 AC DD 001B5 21$: PUSHL 4(R12) ; 1265
                                EF 9F 001B8      PUSHAB INPUT_DESC
00000000G EF 00000000G 02 FB 001BE      CALLS #2,OTSSCVT_T1_L
00000000G EF 00V0000000G 50 D0 001C5      MOVL R0,ISTATUS
                                EF E8 001CC      BLBS ISTATUS,24$ ; 1270
                                00 DD 001D3      PUSHL #0 ; 1274
                                00 DD 001D5      PUSHL #0
                                00 DD 001D7      PUSHL #0
                                8F DD 001D9      PUSHL #11763760
00000000G EF 00B38030 04 FB 001DF      CALLS #4,LIB$SIGNAL
00000000G EF 04 BC D0 001E6 24$: MOVL @4(R12),INPUT_VALUE ; 1280
00000000G EF 04 BC D0 001EE      MOVL @4(R12),INPUT_NUMBER ; 1281
                                04 001F6      RET ; 1283
```

; Routine Size: 503 bytes, Routine Base: \$CODE + 008AA

```

                                0000 MAKE_SCRATCH: ; 1329
                                003C 00000 .WORD ^M<R2,R3,R4,R5>
                                SE C0 AE 9E 00002 MOVAB -64(SP),SP
                                00000040 8F DD 00006 PUSHL #64 ; 1336
00000000G EF 01 FB 0000C CALLS #1,PASS$NEW2
00000000G EF 50 D0 00013 MOVL R0,DEF_SCRATCH
                                C0 AD 00000000G EF D0 0001A MOVL DEF_SCRATCH,-64(FP) ; 1341
C0 BD 00000000G EF 0040 8F 28 00022 MOVC3 #64,LINE_OBJECT_TEMPLATE,@-64(FP)
                                04 0002D      RET ; 1343
```

; Routine Size: 46 bytes, Routine Base: \$CODE + 00AA1

```

                                0000 CURRENT_GT_TEST: ; 1390
                                003C 00000 .WORD ^M<R2,R3,R4,R5>
C0 AD 04 SE C0 AE 9E 00002 MOVAB -64(SP),SP
                                BC 0040 8F 28 00006 MOVC3 #64,@4(R12),TEST
                                5C 08 BC 90 0000E MOVB @8(R12),EXACT_COMPARISON
                                51 00000000G EF 50 94 00012 CLRB CURRENT_GT_TEST ; 1397
                                51 61 9A 0001C ADDL3 #25,DEF_CURRENT,R1 ; 1402
                                51 00000000GEF41 9E 0001F MOVZBL (R1),R1
                                52 D9 AD 9A 00027 MOVAB PRI_SEQ[R1],R1
                                52 00000000GEF42 9E 0002B MOVZBL TEST+25,R2
                                53 00000000G EF D0 00033 MOVAB PRI_SEQ[R2],R2
                                53 1A A3 D0 0003A MOVL DEF_CURRENT,R3
                                00V 5C E9 0003E MOVL 26(R3),R3
                                62 61 91 00041 BLBC EXACT_COMPARISON,16$
                                00V 1B 00044 CMPB (R1),(R2) ; 1406
                                50 01 90 00046 BLEQU 3$
                                DA AD 53 D1 00049 MOVB #1,CURRENT_GT_TEST ; 1410
                                62 00V 15 0004D CMPL R3,TEST+26 ; 1412
                                50 01 90 00052 BLEQ 6$
                                DA AD 53 D1 00057 CMPB (R1),(R2)
                                62 00V 12 00052 BNEQ 6$
                                50 01 90 00054 MOVB #1,CURRENT_GT_TEST ; 1418
                                DA AD 53 D1 00057 CMPL R3,TEST+26 ; 1420
                                62 00V 12 0005B BNEQ 10$
                                5C 00000000G EF D0 00062 CMPB (R1),(R2)
                                DF AD 1F AC D1 00069 MOVL DEF_CURRENT,R12
                                00V 15 0006E CMPL 31(R12),TEST+31
                                BLEQ 10$
```


	50		01	90	00070		MOVB	#1,CURRENT_GT_TEST		: 1428
	5C	00000000G	EF	D0	00073	10\$:	MOVL	DEF CURRENT_RT2		: 1430
DE	AD	1E	AC	91	0007A		CMPB	30(R12),TEST+30		
			00V	1B	0007F		BLEQU	22\$		
DA	AD		53	D1	00081		CMPL	R3,TEST+26		
			00V	12	00085		BNEQ	22\$		
	62		61	91	00087		CMPB	(R1),(R2)		
			00V	12	0008A		BNEQ	22\$		
DF	5C	00000000G	EF	D0	0008C		MOVL	DEF CURRENT_R12		
	AD	1F	AC	D1	00093		CMPL	31(R12),TEST+31		
			00V	12	00098		BNEQ	22\$		
	50		01	90	0009A		MOVB	#1,CURRENT_GT_TEST		: 1440
			00V	11	0009D		BRB	22\$		
	62		61	91	0009F	16\$:	CMPB	(R1),(R2)		: 1448
			00V	1B	000A2		BLEQU	18\$		
DA	50		01	90	000A4		MOVB	#1,CURRENT_GT_TEST		: 1452
	AD		53	D1	000A7	18\$:	CMPL	R3,TEST+26		: 1454
			00V	15	000AB		BLEQ	22\$		
	62		61	91	000AD		CMPB	(R1),(R2)		
			00V	12	000B0		BNEQ	22\$		
	50		01	90	000B2		MOVB	#1,CURRENT_GT_TEST		: 1460
			04	000B5	22\$:		RET			: 1464

; Routine Size: 182 bytes, Routine Base: \$CODE + 00ACF

					00000	CURRENT_LT_TEST:				: 1511
				003C	00000	.WORD	^M<R2,R3,R4,R5>			
			5E	C0	AE	9E	00002	MOVAB	-64(SP),SP	
CO	AD	04	BC	0040	8F	28	00006	MOVC3	#64,24(R12),TEST	
			5C	08	BC	90	0000E	MOVB	28(R12),EXACT_COMPARISON	
					50	94	00012	CLRB	CURRENT_LT_TEST	: 1518
51	00000000G		EF		19	C1	00014	ADDL3	#25,DEF_CURRENT,R1	: 1523
			51		61	9A	0001C	MOVZBL	(R1),R1	
			51	00000000GEF	41	9E	0001F	MOVAB	PRI_SEQ[R1],R1	
			52	D9	AD	9A	00027	MOVZBL	TEST+25,R2	
			52	00000000GEF	42	9E	0002B	MOVAB	PRI_SEQ[R2],R2	
			53	00000000G	EF	D0	00033	MOVL	DEF_CURRENT,R3	
			53	1A	A3	D0	0003A	MOVL	26(R3),R3	
			00V		5C	E9	0003E	BLBC	EXACT_COMPARISON,16\$	
			62		61	91	00041	CMPB	(R1),(R2)	: 1527
					00V	1E	00044	BGEQU	3\$	
			50		01	90	00046	MOVB	#1,CURRENT_LT_TEST	: 1531
DA	AD		53	D1	00049	3\$:	CMPL	R3,TEST+26		: 1533
			00V	18	0004D		BGEQ	6\$		
	62		61	91	0004F		CMPB	(R1),(R2)		
			00V	12	00052		BNEQ	6\$		
DA	50		01	90	00054		MOVB	#1,CURRENT_LT_TEST		: 1539
	AD		53	D1	00057	6\$:	CMPL	R3,TEST+26		: 1541
			00V	12	0005B		BNEQ	10\$		
	62		61	91	0005D		CMPB	(R1),(R2)		
			00V	12	00060		BNEQ	10\$		
DF	5C	00000000G	EF	D0	00062		MOVL	DEF CURRENT_R12		
	AD	1F	AC	D1	00069		CMPL	31(R12),TEST+31		
			00V	18	0006E		BGEQ	10\$		
	50		01	90	00070		MOVB	#1,CURRENT_LT_TEST		: 1549
DE	5C	00000000G	EF	D0	00073	10\$:	MOVL	DEF CURRENT_RT2		: 1551
	AD	1E	AC	91	0007A		CMPB	30(R12),TEST+30		

DA	AD	00V	1E	0007F	BGEQU	22\$	
		53	D1	00081	CMPL	R3,TEST+26	
	62	00V	12	00085	BNEQ	22\$	
		61	91	00087	CMPB	(R1),(R2)	
		00V	12	0008A	BNEQ	22\$	
DF	5C	00000000G	EF	D0	0008C	MOVL	DEF,CURRENT,R12
	AD	1F	AC	D1	00093	CMPL	31(R12),TEST+31
			00V	12	00098	BNEQ	22\$
	50		01	90	0009A	MOVB	#1,CURRENT_LT_TEST
			00V	11	0009D	BRB	22\$
	62		61	91	0009F	CMPB	(R1),(R2)
			00V	1E	000A2	BGEQU	18\$
	50		01	90	000A4	MOVB	#1,CURRENT_LT_TEST
DA	AD		53	D1	000A7	CMPL	R3,TEST+26
			00V	18	000AB	BGEQ	22\$
	62		61	91	000AD	CMPB	(R1),(R2)
			00V	12	000B0	BNEQ	22\$
	50		01	90	000B2	MOVB	#1,CURRENT_LT_TEST
			04	000B5	22\$:	RET	

; Routine Size: 182 bytes, Routine Base: \$CODE + 00B85

				00000	CURRENT_EQ_TEST:		
			003C	00000	.WORD	^M<R2,R3,R4,R5>	: 1632
			AE	9E	00002	MOVAB	-64(SP),SP
			8F	28	00006	MOVBC3	#64,a4(R12),TEST
			BC	90	0000E	MOVB	a8(R12),EXACT_COMPARISON
			19	C1	00012	ADDL3	#25,DEF_CURRENT,R0
			51	94	0001A	CLRB	R1
			AD	91	0001C	CMPB	TEST+25,(R0)
			00V	12	00020	BNEQ	2\$
			51	96	00022	INCB	R1
			1A	C1	00024	2\$:	ADDL3
			52	94	0002C	CLRB	R2
			AD	D1	0002E	CMPL	TEST+26,(R0)
			00V	12	00032	BNEQ	4\$
			52	96	00034	INCB	R2
			5C	E9	00036	4\$:	BLBC
			EF	D0	00039	MOVL	DEF_CURRENT,R12
			AD	D1	00040	CMPL	TEST+31,31(R12)
			00V	12	00045	BNEQ	11\$
			EF	D0	00047	MOVL	DEF_CURRENT,R12
			AD	91	0004E	CMPB	TEST+30,30(R12)
			00V	12	00053	BNEQ	11\$
			52	E9	00055	BLBC	R2,11\$
			EF	D0	00058	MOVL	DEF_CURRENT,R12
			AD	91	0005F	CMPB	TEST,(R12)
			00V	12	00063	BNEQ	11\$
			51	E9	00065	BLBC	R1,11\$
			01	90	00068	MOVB	#1,CURRENT_EQ_TEST
			00V	11	0006B	BRB	14\$
			5C	94	0006D	11\$:	CLRB
			00V	11	0006F	BRB	14\$
			52	92	00071	13\$:	MCOMB
			52	8B	00074	BICB3	R2,R1,CURRENT_EQ_TEST
			5C	90	00078	14\$:	MOVB
			04	0007B	RET		

; Routine Size: 124 bytes, Routine Base: \$CODE + 00C3B

			0000	00000	INSERT_BEFORE_CURRENT:		: 1716
			0000	00000	.WORD	"M<>	
00000000G	EF	00000000G	EF	D1 00002	CMPL	DEF_CURRENT,DEF_HEAD	: 1723
			00V	12 0000D	BNEQ	2\$	
00000000G	EF	00000000G	EF	D0 0000F	MOVL	DEF_SCRATCH,DEF_HEAD	: 1725
00000000G	50	00000000G	EF	D0 0001A	2\$:	MOVL	DEF_CURRENT,R0
	EF	05	A0	D0 00021	MOVL	5(R0),DEF_PRED	: 1730
00000000G	50	00000000G	EF	D0 00029	MOVL	DEF_SCRATCH,R0	: 1731
01	A0	00000000G	EF	D0 00030	MOVL	DEF_CURRENT,1(R0)	
	50	00000000G	EF	D0 00038	MOVL	DEF_SCRATCH,R0	: 1732
05	A0	00000000G	EF	D0 0003F	MOVL	DEF_PRED,5(R0)	
		00000000G	EF	D5 00047	TSTL	DEF_PRED	: 1734
			00V	13 0004D	BEQL	4\$	
	50	00000000G	EF	D0 0004F	MOVL	DEF_PRED,R0	: 1736
01	A0	00000000G	EF	D0 00056	MOVL	DEF_SCRATCH,1(R0)	
	50	00000000G	EF	D0 0005E	4\$:	MOVL	DEF_CURRENT,R0
05	A0	00000000G	EF	D0 00065	MOVL	DEF_SCRATCH,5(R0)	: 1738
00000000G	EF	00000000G	EF	D0 0006D	MOVL	DEF_SCRATCH,DEF_CURRENT	: 1743
			04	00078	RET		: 1745

; Routine Size: 121 bytes, Routine Base: \$CODE + 00CB7

			0000	00000	INSERT_AT_CURRENT:		: 1793
			0000	00000	.WORD	"M<>	
00000000G	EF	00000000G	EF	D1 00002	CMPL	DEF_CURRENT,DEF_HEAD	: 1800
			00V	12 0000D	BNEQ	2\$	
00000000G	EF	00000000G	EF	D0 0000F	MOVL	DEF_SCRATCH,DEF_HEAD	: 1802
00000000G	EF	00000000G	EF	D1 0001A	2\$:	CMPL	DEF_CURRENT,DEF_TAIL
			00V	12 00025	BNEQ	4\$: 1804
00000000G	EF	00000000G	EF	D0 00027	MOVL	DEF_SCRATCH,DEF_TAIL	: 1806
	50	00000000G	EF	D0 00032	4\$:	MOVL	DEF_CURRENT,R0
00000000G	EF	05	A0	D0 00039	MOVL	5(R0),DEF_PRED	: 1811
	50	00000000G	EF	D0 00041	MOVL	DEF_CURRENT,R0	: 1812
00000000G	EF	01	A0	D0 00048	MOVL	1(R0),DEF_SUCC	
	50	00000000G	EF	D0 00050	MOVL	DEF_SCRATCH,R0	: 1813
	51	00000000G	EF	D0 00057	MOVL	DEF_CURRENT,R1	
01	A0	01	A1	D0 0005E	MOVL	1(RT),1(R0)	
	50	00000000G	EF	D0 00063	MOVL	DEF_SCRATCH,R0	: 1814
	51	00000000G	EF	D0 0006A	MOVL	DEF_CURRENT,R1	
05	A0	05	A1	D0 00071	MOVL	5(RT),5(R0)	
		00000000G	EF	D5 00076	TSTL	DEF_PRED	: 1816
			00V	13 0007C	BEQL	6\$	
	50	00000000G	EF	D0 0007E	MOVL	DEF_PRED,R0	: 1818
01	A0	00000000G	EF	D0 00085	MOVL	DEF_SCRATCH,1(R0)	
		00000000G	EF	D5 0008D	6\$:	TSTL	DEF_SUCC
			00V	13 00093	BEQL	8\$: 1820
	50	00000000G	EF	D0 00095	MOVL	DEF_SUCC,R0	: 1822
05	A0	00000000G	EF	D0 0009C	MOVL	DEF_SCRATCH,5(R0)	
		00000000G	EF	DD 000A4	8\$:	PUSHL	DEF_CURRENT
00000000G	EF		01	FB 000AA	CALLS	#1,PASS\$DISPOSE2	: 1827
00000000G	EF	00000000G	EF	D0 000B1	MOVL	DEF_SCRATCH,DEF_CURRENT	: 1829
			04	000BC	RET		: 1831

; Routine Size: 189 bytes, Routine Base: \$CODE + 00D30


```
00000000G EF 00000000G EF 0000 000000 INSERT_AFTER CURRENT: ; 1879
00000000G EF 00000000G EF 0000 000000 .WORD ^M<> ; 1886
00000000G EF 00000000G EF 0000 000002 CMPL DEF_CURRENT,DEF_TAIL ; 1888
00000000G EF 00000000G EF 0000 00000D BNEQ 2$ DEF_SCRATCH,DEF_TAIL ; 1893
00000000G EF 00000000G EF 0000 00000F MOVL DEF_CURRENT,R0 ; 1894
00000000G EF 00000000G EF 0000 00001A 2$: MOVL 1(R0),DEF_SUCC ; 1895
00000000G EF 00000000G EF 0000 000021 MOVL DEF_SCRATCH,R0 ; 1896
00000000G EF 00000000G EF 0000 000029 MOVL DEF_SUCC,1(R0) ; 1898
00000000G EF 00000000G EF 0000 000030 MOVL DEF_SCRATCH,R0 ; 1900
00000000G EF 00000000G EF 0000 000038 MOVL DEF_CURRENT,5(R0) ; 1905
00000000G EF 00000000G EF 0000 00003F MOVL DEF_CURRENT,R0 ; 1907
00000000G EF 00000000G EF 0000 000047 MOVL DEF_SCRATCH,1(R0) ; 1907
00000000G EF 00000000G EF 0000 00004E TSTL DEF_SUCC ; 1907
00000000G EF 00000000G EF 0000 000056 BEQL 4$ ; 1907
00000000G EF 00000000G EF 0000 00005C MOVL DEF_SUCC,R0 ; 1907
00000000G EF 00000000G EF 0000 00005E MOVL DEF_SCRATCH,5(R0) ; 1907
00000000G EF 00000000G EF 0000 000065 MOVL DEF_SCRATCH,DEF_CURRENT ; 1907
00000000G EF 00000000G EF 0000 00006D 4$: RET ; 1907
00000000G EF 00000000G EF 0000 000078
```

; Routine Size: 121 bytes, Routine Base: \$CODE + 00DED

```
00000000G EF 00000000G EF 0000 000000 INCR_CURRENT: ; 1953
00000000G EF 00000000G EF 0000 000000 .WORD ^M<> ; 1960
00000000G EF 00000000G EF 0000 000002 TSTL DEF_CURRENT ; 1960
00000000G EF 00000000G EF 0000 000008 BEQL 2$ ; 1962
00000000G EF 00000000G EF 0000 00000A MOVL DEF_CURRENT,R0 ; 1962
00000000G EF 00000000G EF 0000 000011 MOVL 1(R0),DEF_CURRENT ; 1964
00000000G EF 00000000G EF 0000 000019 2$: RET ; 1964
```

; Routine Size: 26 bytes, Routine Base: \$CODE + 00E66

```
00000000G EF 00000000G EF 0000 000000 DECR_CURRENT: ; 2010
00000000G EF 00000000G EF 0000 000000 .WORD ^M<> ; 2017
00000000G EF 00000000G EF 0000 000002 TSTL DEF_CURRENT ; 2017
00000000G EF 00000000G EF 0000 000008 BEQL 2$ ; 2019
00000000G EF 00000000G EF 0000 00000A MOVL DEF_CURRENT,R0 ; 2019
00000000G EF 00000000G EF 0000 000011 MOVL 5(R0),DEF_CURRENT ; 2021
00000000G EF 00000000G EF 0000 000019 2$: RET ; 2021
```

; Routine Size: 26 bytes, Routine Base: \$CODE + 00E80

```
00000000G EF 00000000G EF 0000 000000 NEW_IDENT LINE: ; 2075
00000000G EF 00000000G EF 0000 000000 .WORD ^M<R2,R3> ; 2086
00000000G EF 00000000G EF 0000 000002 SUBL2 #28,SP ; 2091
00000000G EF 00000000G EF 0000 000005 CALLS #0,MAKE_SCRATCH ; 2091
00000000G EF 00000000G EF 0000 00000A MOVL #17694740,-8(FP) ; 2091
00000000G EF 00000000G EF 0000 000012 MOVAB DATE_STRING,-4(FP) ; 2091
00000000G EF 00000000G EF 0000 000017 PUSHAB -8(FP) ; 2091
00000000G EF 00000000G EF 0000 00001A CALLS #1,LIB$DATE_TIME ; 2096
00000000G EF 00000000G EF 0000 000021 MOVL #1,R0 ; 2096
00000000G EF 00000000G EF 0000 000024 2$: MOVL R0,I ; 2098
00000000G EF 00000000G EF 0000 000027 MOVAB DATE_STRING-1[I],IDENT_STRING-1[I] ; 2098
00000000G EF 00000000G EF 0000 000031 AOBLEQ #20,R0,2$ ; 2103
00000000G EF 00000000G EF 0000 000035 MOVL DEF_SCRATCH,R2 ; 2107
00000000G EF 00000000G EF 0000 00003C MOVQ NULL_STRING,TEMP_DESCRIPTOR ; 2107
```


00000000G	EF	000000FF	8F	DD	00047	PUSHL	#255	: 2108
00000004G	EF		01	FB	0004D	CALLS	#1,PASS\$NEW2	
00000000G	EF	00000000G	50	D0	00054	MOVL	R0,TEMP_DESCRIPTOR+4	
			EF	B0	0005B	MOVW	IDENT_STRING_LENGTH,TEMP_DESCRIPTOR	: 2110
			62	94	00066	CLRB	(R2)	: 2111
19	A2		09	90	00068	MOVB	#9,25(R2)	: 2112
	50		01	D0	0006C	MOVL	#1,R0	: 2114
	51	00000000G	EF	D0	0006F	MOVL	IDENT_STRING_LENGTH,R1	
	51		50	D1	00076	CMPL	R0,R1	
			00V	15	00079	BLEQ	5\$	
			00V	11	0007B	BRB	6\$	
			50	D6	0007D	INCL	R0	
	5C		50	D0	0007F	MOVL	R0,I	
	53	00000004G	EF	D0	00082	MOVL	TEMP_DESCRIPTOR+4,R3	: 2116
FF A34C	FF	FFFFFFFFG	EF	90	00089	MOVB	IDENT_STRING-1[I],-1(R3)[I]	
	51		50	D1	00093	CMPL	R0,R1	
			E5	19	00096	BLSS	4\$	
		11	A2	9F	00098	PUSHAB	17(R2)	: 2118
		00000000G	EF	9F	0009B	PUSHAB	TEMP_DESCRIPTOR	
00000000G	EF	00000000G	02	FB	000A1	CALLS	#2,LIB\$SCOPY_DXDX	
		00000004G	EF	DD	000A8	PUSHL	TEMP_DESCRIPTOR+4	: 2119
00000000G	EF		01	FB	000AE	CALLS	#1,PASS\$DISPOSE2	
00000000G	EF	00000000G	EF	D0	000B5	MOVL	DEF_SCRATCH,DEF_CURRENT	: 2126
00000000G	EF	00000000G	EF	D0	000C0	MOVL	DEF_SCRATCH,DEF_HEAD	: 2127
00000000G	EF	00000000G	EF	D0	000CB	MOVL	DEF_SCRATCH,DEF_TAIL	: 2128
			04	000D6	RET			: 2130

; Routine Size: 215 bytes, Routine Base: \$CODE + 00E9A

				00000	DELETE_CURRENT:		: 2180	
			0000	00000	.WORD	^M<>		
	50	00000000G	EF	D0	00002	MOVL	DEF_CURRENT,R0	: 2184
	0F	19	A0	91	00009	CMPB	25(R0),#15	
			00V	12	0000D	BNEQ	5\$	
	50	00000000G	EF	D0	0000F	MOVL	DEF_CURRENT,R0	: 2191
		01	A0	D5	00016	TSTL	1(R0)	
			00V	12	00019	BNEQ	3\$	
		00000000G	EF	DD	0001B	PUSHL	DEF_CURRENT	: 2195
00000000G	EF		01	FB	00021	CALLS	#1,PASS\$DISPOSE2	
0E9A	CF		00	FB	00028	CALLS	#0,NEW_IDENT_LINE	: 2196
			0000V	31	0002D	BRW	27\$	
	50	00000000G	EF	D0	00030	MOVL	DEF_CURRENT,R0	: 2204
00000000G	EF	01	A0	D0	00037	MOVL	1(R0),DEF_HEAD	
	50	00000000G	EF	D0	0003F	MOVL	DEF_HEAD,R0	: 2205
		05	A0	D4	00046	CLRL	5(R0)	
		00000000G	EF	DD	00049	PUSHL	DEF_CURRENT	: 2206
00000000G	EF		01	FB	0004F	CALLS	#1,PASS\$DISPOSE2	
00000000G	EF	00000000G	EF	D0	00056	MOVL	DEF_HEAD,DEF_CURRENT	: 2207
			0000V	31	00061	BRW	27\$	
		00000000G	EF	D5	00064	TSTL	DEF_CURRENT	: 2220
			00V	13	0006A	BEQL	8\$	
00000000G	EF	00000000G	EF	D1	0006C	CMPL	DEF_CURRENT,DEF_TAIL	
			00V	12	00077	BNEQ	8\$	
	50	00000000G	EF	D0	00079	MOVL	DEF_CURRENT,R0	: 2222
00000000G	EF	05	A0	D0	00080	MOVL	5(R0),DEF_TAIL	
		00000000G	EF	D5	00088	TSTL	DEF_CURRENT	: 2227
			00V	13	0008E	BEQL	11\$	

Generated Code		5-Sep-1984 13:38:55		DISK\$VMSMASTER:[EDF.SRC]EDFUTIL.PAS;1 (59)	
00000000G	EF	00000000G	EF	D1 00090	CMPL DEF_CURRENT,DEF_HEAD
			00V	12 0009B	BNEQ 11\$
	50	00000000G	EF	D0 0009D	MOVL DEF_CURRENT,R0
00000000G	EF	01	A0	D0 000A4	MOVL 1(R0),DEF_HEAD
	50	00000000G	EF	D0 000AC	MOVL DEF_CURRENT,R0
00000000G	EF	05	A0	D0 000B3	MOVL 5(R0),DEF_PRED
	50	00000000G	EF	D0 000BB	MOVL DEF_CURRENT,R0
00000000G	EF	01	A0	D0 000C2	MOVL 1(R0),DEF_SUCC
		00000000G	EF	D5 000CA	TSTL DEF_PRED
			00V	13 000D0	BEQL 13\$
	50	00000000G	EF	D0 000D2	MOVL DEF_PRED,R0
01	A0	00000000G	EF	D0 000D9	MOVL DEF_SUCC,1(R0)
		00000000G	EF	D5 000E1	TSTL DEF_SUCC
			00V	13 000E7	BEQL 15\$
	50	00000000G	EF	D0 000E9	MOVL DEF_SUCC,R0
05	A0	00000000G	EF	D0 000F0	MOVL DEF_PRED,5(R0)
	5C	00000000G	EF	D0 000F8	MOVL DEF_CURRENT,R12
		11	AC	B5 000FF	TSTW 17(R12)
			00V	15 00102	BLEQ 18\$
		11	AC	9F 00104	PUSHAB 17(R12)
00000000G	EF		01	FB 00107	CALLS #1,STR\$FREE1_DX
		09	AC	B5 0010E	TSTW 9(R12)
			00V	15 00111	BLEQ 20\$
		09	AC	9F 00113	PUSHAB 9(R12)
00000000G	EF		01	FB 00116	CALLS #1,STR\$FREE1_DX
		00000000G	EF	DD 0011D	PUSHL DEF_CURRENT
00000000G	EF		01	FB 00123	CALLS #1,PASS\$DISPOSE2
		00000000G	EF	D5 0012A	TSTL DEF_SUCC
			00V	13 00130	BEQL 22\$
00000000G	EF	00000000G	EF	D0 00132	MOVL DEF_SUCC,DEF_CURRENT
			00V	11 0013D	BRB 26\$
		00000000G	EF	D5 0013F	TSTL DEF_PRED
			00V	13 00145	BEQL 24\$
00000000G	EF	00000000G	EF	D0 00147	MOVL DEF_PRED,DEF_CURRENT
			00V	11 00152	BRB 25\$
0E9A	CF		00	FB 00154	CALLS #0,NEW_IDENT_LINE
				00159	
				00159	
				00159	
			04	00159	RET

; Routine Size: 346 bytes, Routine Base: \$CODE + 00F71

Address	Op-Code	Op-Code Hex	Op-Code Bin	Op-Code Dec	Op-Code Name	Op-Code Comment	Op-Code Address
00000000G	52	04	BC	001C	00000	DELETE_PRIMARY_SECTION:	: 2327
	5C	08	BC	90	00002	.WORD	
	EF	00000000G	EF	D0	00006	MOVW	
			53	94	00015	MOVL	
			54	94	00017	MOVL	
	50	00000000G	EF	D0	00019	DEF HEAD,DEF_CURRENT	: 2336
	5C	1A	A0	D1	00020	DOING	: 2337
			00V	12	00024	CLRB	: 2338
	50	00000000G	EF	D0	00026	CLRB	: 2348
			60	95	0002D	DONE	
			00V	12	0002F	MOVW	
	50	00000000G	EF	D0	00031	DEF CURRENT,RO	
	52	19	A0	91	00038	26(RO),WHICHPRINUM	
						BNEQ	
						5\$	
						DEF CURRENT,RO	
						(RO)	
						TSTB	
						5\$	
						DEF CURRENT,RO	
						25(RO),WHICHPRIMARY	

Generated Code

```
00V 12 0003C BNEQ 5$
01 90 0003E MOVB #1,DOING ; 2356
53 E9 00041 5$: BLBC DOING,7$ ; 2361
OF71 CF 00 FB 00044 CALLS #0,DELETE_CURRENT ; 2363
00V 11 00049 BRB 8$
0E66 CF 00 FB 0004B 7$: CALLS #0,INCR_CURRENT ; 2370
00000000G EF D5 00050 8$: TSTL DEF_CURRENT ; 2376
00V 13 00056 BEQL 15$
00V 53 E9 00058 BLBC DOING,15$ ; 2378
50 00000000G EF D0 0005B MOVL DEF_CURRENT,R0
60 95 00062 TSTB (R0)
00V 12 00064 BNEQ 15$
50 00000000G EF D0 00066 MOVL DEF_CURRENT,R0
5C 1A A0 D1 0006D CMPL 26(R0),WHICHPRINUM
00V 12 00071 BNEQ 13$
50 00000000G EF D0 00073 MOVL DEF_CURRENT,R0
52 19 A0 91 0007A CMPB 25(R0),WHICHPRIMARY
00V 13 0007E BEQL 15$
54 01 90 00080 13$: MOVB #1,DONE ; 2390
00V 54 E8 00083 15$: BLBS DONE,17$
00000000G EF D5 00086 TSTL DEF_CURRENT
8B 12 0008C BNEQ 1$
04 0008E 17$: RET ; 2394
```

; Routine Size: 143 bytes, Routine Base: \$CODE + 010CB

```
00000000G EF 00000000G EF 0000 00000 INIT_DEF: ; 2441
0000 00000 .WORD ^M<>
00V 13 0000D MOVL DEF_HEAD,DEF_CURRENT ; 2448
OF71 CF 00 FB 0000F 2$: BEQL 6$ ; 2450
00000000G EF 00000000G EF D1 00014 CALLS #0,DELETE_CURRENT ; 2456
EE 12 0001F CMPL DEF_HEAD,DEF_TAIL
00000000G EF D5 00021 BNEQ 2$
00V 13 00027 TSTL DEF_CURRENT ; 2460
OF71 CF 00 FB 00029 BEQL 6$
04 0002E 6$: CALLS #0,DELETE_CURRENT ; 2462
RET ; 2466
```

; Routine Size: 47 bytes, Routine Base: \$CODE + 0115A

```
00000000G EF 00000000G EF 0000 00000 INSERT_IN_ORDER: ; 2516
001C 00000 .WORD ^M<R2,R3,R4>
5C 04 BC D0 00002 MOVL @4(R12),COLLISION_ACTION ; 2528
50 00000000G EF D0 00006 MOVL DEF_SCRATCH,R0
60 95 0000D TSTB (R0)
00V 12 0000F BNEQ 3$
50 00000000G EF D0 00011 MOVL DEF_SCRATCH,R0
OF 19 A0 91 00018 CMPB 25(R0),#15
00V 13 0001C BEQL 3$
50 00000000G EF D0 0001E MOVL DEF_SCRATCH,R0 ; 2534
11 A0 B4 00025 CLRW 17(R0)
50 00000000G EF D0 00028 3$: MOVL DEF_SCRATCH,R0 ; 2536
01 A0 D4 0002F CLRL 1(R0)
50 00000000G EF D0 00032 MOVL DEF_SCRATCH,R0 ; 2537
05 A0 D4 00039 CLRL 5(R0)
00000000G 52 94 0003C CLRB BACKUP_WORKED ; 2542
EF D5 0003E TSTL DEF_CURRENT ; 2544
```


		03	12	00044	BNEQ	+3		
		0000V	31	00046	BRW	14\$		
	50	00000000G	EF	D0	00049	MOVL	DEF_CURRENT,R0	: 2546
		05	A0	D5	00050	TSTL	5(R0)	
			03	12	00053	BNEQ	+3	
		0000V	31	00055	BRW	14\$		
0E80	CF		00	FB	00058	CALLS	#0,DECR_CURRENT	: 2550
			00V	11	0005D	BRB	7\$: 2552
0E66	CF		00	FB	0005F	CALLS	#0,INCR_CURRENT	: 2560
		01	8F	9F	00064	PUSHAB	#1	
	50	00000000G	EF	D0	00067	MOVL	DEF_SCRATCH,R0	
			60	9F	0006E	PUSHAB	(R0)	
0ACF	CF		02	FB	00070	CALLS	#2,CURRENT_GT_TEST	
	53		50	90	00075	MOVB	R0,R3	
		01	8F	9F	00078	PUSHAB	#1	
	50	00000000G	EF	D0	0007B	MOVL	DEF_SCRATCH,R0	
			60	9F	00082	PUSHAB	(R0)	
0C3B	CF		02	FB	00084	CALLS	#2,CURRENT_EQ_TEST	
	50		53	88	00089	BISB2	R3,R0	
			53	94	0008C	CLRB	R3	
	51	00000000G	EF	D0	0008E	MOVL	DEF_CURRENT,R1	
		01	A1	D5	00095	TSTL	1(RT)	
			00V	12	00098	BNEQ	9\$	
			53	96	0009A	INCB	R3	
	53		50	88	0009C	BISB2	R0,R3	
	BD		53	E9	0009F	BLBC	R3,6\$	
		01	8F	9F	000A2	PUSHAB	#1	: 2562
	50	00000000G	EF	D0	000A5	MOVL	DEF_SCRATCH,R0	
			60	9F	000AC	PUSHAB	(R0)	
0C3B	CF		02	FB	000AE	CALLS	#2,CURRENT_EQ_TEST	
	53		50	90	000B3	MOVB	R0,R3	
		01	8F	9F	000B6	PUSHAB	#1	
	50	00000000G	EF	D0	000B9	MOVL	DEF_SCRATCH,R0	
			60	9F	000C0	PUSHAB	(R0)	
0B85	CF		02	FB	000C2	CALLS	#2,CURRENT_LT_TEST	
			51	94	000C7	CLRB	R1	
	54	00000000G	EF	D0	000C9	MOVL	DEF_CURRENT,R4	
		01	A4	D5	000D0	TSTL	1(R4)	
			00V	13	000D3	BEQL	12\$	
			51	96	000D5	INCB	R1	
	50		51	8A	000D7	BICB2	R1,R0	12\$:
52	53		50	89	000DA	BISB3	R0,R3,BACKUP_WORKED	
	00V		52	E8	000DE	BLBS	BACKUP_WORKED,21\$: 2574
	00000000G	EF	00000000G	D0	000E1	MOVL	DEF_HEAD,DEF_CURRENT	: 2581
			00V	11	000EC	BRB	17\$: 2583
	0E66	CF	00	FB	000EE	CALLS	#0,INCR_CURRENT	: 2591
		01	8F	9F	000F3	PUSHAB	#1	
	52	00000000G	EF	D0	000F6	MOVL	DEF_SCRATCH,R2	
			62	9F	000FD	PUSHAB	(R2)	
0ACF	CF		02	FB	000FF	CALLS	#2,CURRENT_GT_TEST	
	52		50	90	00104	MOVB	R0,R2	
		01	8F	9F	00107	PUSHAB	#1	
	50	00000000G	EF	D0	0010A	MOVL	DEF_SCRATCH,R0	
			60	9F	00111	PUSHAB	(R0)	
0C3B	CF		02	FB	00113	CALLS	#2,CURRENT_EQ_TEST	
	50		52	88	00118	BISB2	R2,R0	
			52	94	0011B	CLRB	R2	

Generated Code			
53	00000000G	EF	D0 0011D
	01	A3	D5 00124
		00V	12 00127
52		52	96 00129
BD		50	88 0012B 19\$:
	01	52	E9 0012E
		8F	9F 00131 21\$:
50	00000000G	EF	D0 00134
		60	9F 0013B
OACF	CF	02	FB 0013D
	00V	50	E9 00142
OCB7	CF	00	FB 00145
		00V	11 0014A
	01	8F	9F 0014C 23\$:
50	00000000G	EF	D0 0014F
		60	9F 00156
OC3B	CF	02	FB 00158
	00V	50	E9 0015D
		5C	D5 00160
		00V	12 00162
OD30	CF	00	FB 00164
		00V	11 00169
	02	5C	D1 0016B 26\$:
		00V	12 0016E
ODED	CF	00	FB 00170
		00V	11 00175
	50	EF	D0 00177 30\$:
	01	A0	D5 0017E
		00V	12 00181
00000000G	EF	00000000G	EF D0 00183
ODED	CF	00	FB 0018E
		04	00193 34\$:
			MOV L DEF_CURRENT,R3
			TSTL 1(R3)
			BNEQ 19\$
			INCB R2
			BISB2 R0,R2
			BLBC R2,16\$
			PUSHAB #1 ; 2598
			MOV L DEF_SCRATCH,R0
			PUSHAB (R0)
			CALLS #2,CURRENT_GT_TEST
			BLBC R0,23\$
			CALLS #0,INSERT_BEFORE_CURRENT ; 2600
			BRB 34\$
			PUSHAB #1 ; 2602
			MOV L DEF_SCRATCH,R0
			PUSHAB (R0)
			CALLS #2,CURRENT_EQ_TEST
			BLBC R0,30\$
			TSTL COLLISION_ACTION ; 2606
			BNEQ 26\$
			CALLS #0,INSERT_AT_CURRENT ; 2608
			BRB 34\$
			CMPL COLLISION_ACTION,#2 ; 2610
			BNEQ 34\$
			CALLS #0,INSERT_AFTER_CURRENT ; 2612
			BRB 34\$
			MOV L DEF_CURRENT,R0 ; 2618
			TSTL 1(R0)
			BNEQ 34\$
			MOV L DEF_CURRENT,DEF_TAIL ; 2622
			CALLS #0,INSERT_AFTER_CURRENT ; 2623
			RET ; 2627

; Routine Size: 404 bytes, Routine Base: \$CODE + 01189

			00000	FIND_OBJECT:		
			000C	00000	.WORD	
			9E	00002	MOVAB	^M<R2,R3>
			90	00006	MOVB	-64(SP),SP
			90	0000A	MOVB	@4(R12),OBJ_TYP
			D0	0000E	MOVB	@8(R12),PRIM
			90	00012	MOVL	@12(R12),PRIMNUM
			D0	00016	MOVB	@16(R12),SECO
			D0	0001A	MOVL	@20(R12),SECONUM
			90	0001E	MOVB	OBJ_TYP,TEST
			D0	00022	MOVB	PRIM,TEST+25 ; 2696
			90	00026	MOVL	PRIMNUM,TEST+26 ; 2697
			D0	0002A	MOVB	SECO,TEST+30 ; 2698
			EF	0002E	MOVL	SECONUM,TEST+31 ; 2699
			5C	94 00039	MOVL	DEF_HEAD,DEF_CURRENT ; 2700
			53	94 0003B	CLRB	FOUND_IT ; 2705
			EF	D5 0003D	CLRB	PAST_IT ; 2706
			00V	13 00043	TSTL	DEF_CURRENT ; 2707
			8F	9F 00045 2\$:	BEQL	8\$; 2709
			AD	9F 00048	PUSHAB	#1
			02	FB 0004B	PUSHAB	TEST
			50	90 00050	CALLS	#2,CURRENT_EQ_TEST
					MOVB	R0,FOUND_IT

5E	C0	AE	00000	FIND_OBJECT:		
50	04	BC	9E 00002	.WORD		
51	08	BC	90 00006	MOVAB	^M<R2,R3>	
52	0C	BC	90 0000A	MOVB	-64(SP),SP	
53	10	BC	D0 0000E	MOVB	@4(R12),OBJ_TYP	
5C	14	BC	90 00012	MOVL	@8(R12),PRIM	
		BC	D0 00016	MOVB	@12(R12),PRIMNUM	
		BC	90 0001A	MOVL	@16(R12),SECO	
		BC	D0 0001E	MOVB	@20(R12),SECONUM	
		BC	90 00022	MOVB	OBJ_TYP,TEST	
		BC	D0 00026	MOVB	PRIM,TEST+25 ; 2696	
		BC	90 0002A	MOVL	PRIMNUM,TEST+26 ; 2697	
		BC	D0 0002E	MOVB	SECO,TEST+30 ; 2698	
		BC	90 00039	MOVL	SECONUM,TEST+31 ; 2699	
		BC	D0 0003D	MOVL	DEF_HEAD,DEF_CURRENT ; 2700	
		BC	94 0003B	CLRB	FOUND_IT ; 2705	
		BC	D5 0003D	CLRB	PAST_IT ; 2706	
		BC	13 00043	TSTL	DEF_CURRENT ; 2707	
		BC	9F 00045 2\$:	BEQL	8\$; 2709	
		BC	9F 00048	PUSHAB	#1	
		BC	02 FB 0004B	PUSHAB	TEST	
		BC	50 90 00050	CALLS	#2,CURRENT_EQ_TEST	
		BC		MOVB	R0,FOUND_IT	

Generated Code

		01	8F	9F	00053	PUSHAB	#1	: 2716
		CO	AD	9F	00056	PUSHAB	TEST	
OACF	CF		02	FB	00059	CALLS	#2,CURRENT_GT_TEST	
	53		50	90	0005E	MOVB	R0,PAST_IT	
OE66	00V		5C	E8	00061	BLBS	FOUND_IT,4\$: 2718
	CF		00	FB	00064	CALLS	#0,INCR_CURRENT	: 2720
	00V		5C	E8	00069	BLBS	FOUND_IT,8\$	
	00V		53	E8	0006C	BLBS	PAST_IT,8\$	
	00000000G		EF	D5	0006F	TSTL	DEF_CURRENT	
			CE	12	00075	BNEQ	2\$	
	50		5C	90	00077	MOVB	FIND_OBJECT,R0	: 2731
				04	0007A	RET		

; Routine Size: 123 bytes, Routine Base: \$CODE + 0131D

00V00000000G	EF	00000000G	00	0000	00000	.ENTRY	POINT_AT_DEFINITION,^M<>	: 2781
00000000G	EF	00000000G	EF	D0	00002	BBS	#0,POINTING_AT_DEFINITION,2\$: 2785
00000000G	EF	00000000G	EF	D0	0000A	MOVL	DEF_HEAD,DEF_ANL_HEAD	: 2789
00000000G	EF	00000000G	EF	D0	00015	MOVL	DEF_TAIL,DEF_ANL_TAIL	: 2790
00000000G	EF	00000000G	EF	D0	00020	MOVL	DEF_SAVE_HEAD,DEF_HEAD	: 2791
00000000G	EF	00000000G	EF	D0	0002B	MOVL	DEF_SAVE_TAIL,DEF_TAIL	: 2792
00000000G	EF		01	90	00036	MOVB	#1,POINTING_AT_DEFINITION	: 2794
				04	0003D	RET		: 2798

; Routine Size: 62 bytes, Routine Base: \$CODE + 01398

				0000	00000	POINT_AT_ANALYSIS:		: 2848
00V00000000G	EF	00000000G	00	0000	00000	.WORD	^M<>	
00000000G	EF	00000000G	EF	E1	00002	BBC	#0,POINTING_AT_DEFINITION,2\$: 2852
00000000G	EF	00000000G	EF	D0	0000A	MOVL	DEF_HEAD,DEF_SAVE_HEAD	: 2856
00000000G	EF	00000000G	EF	D0	00015	MOVL	DEF_TAIL,DEF_SAVE_TAIL	: 2857
00000000G	EF	00000000G	EF	D0	00020	MOVL	DEF_ANL_HEAD,DEF_HEAD	: 2858
00000000G	EF	00000000G	EF	D0	0002B	MOVL	DEF_ANL_TAIL,DEF_TAIL	: 2859
		00000000G	EF	94	00036	CLRB	POINTING_AT_DEFINITION	: 2861
				04	0003C	RET		: 2865

; Routine Size: 61 bytes, Routine Base: \$CODE + 013D6

				0004	00000	.ENTRY	EDF\$LINE_PARSED,^M<R2>	: 2915
				D0	00002	MOVL	#1,EDF\$LINE_PARSED	: 2922
	0AA1	5C	01	FB	00005	CALLS	#0,MAKE_SCRATCH	: 2927
00000000G	EF	50	00	D0	0000A	MOVL	FDL_BLOCK,R0	: 2932
	18	00	60	F0	00011	INSV	(R0),#0,#24,TEMP_FDL3\$TYPE	
00V00000000G	EF	50	05	E1	0001A	BBC	#5,TEMP_FDL3\$TYPE,2\$: 2937
		09	A0	91	00022	MOVL	FDL_BLOCK,R0	
			03	12	0002D	CMPB	8(R0),#9	
			03	31	0002F	BNEQ	+3	
			03	E1	00032	BRW	76\$	
03 00000000G	EF		03	E1	00032	BBC	#3,TEMP_FDL3\$TYPE,++3	
			03	31	0003A	BRW	76\$	
	52	00000000G	EF	D0	0003D	MOVL	DEF_SCRATCH,R2	: 2945
00V00000000G	EF		05	E1	00044	BBC	#5,TEMP_FDL3\$TYPE,6\$: 2952
			62	94	0004C	CLRB	(R2)	: 2954
			00V	11	0004E	BRB	7\$	
	62		01	90	00050	MOVB	#1,(R2)	: 2958
00V00000000G	EF		09	E1	00053	BBC	#9,TEMP_FDL3\$TYPE,9\$: 2963
	62		02	90	0005B	MOVB	#2,(R2)	: 2965

19	50	00000000G	EF	D0	0005E	9\$:	MOVL	FDL_BLOCK,R0	:	2970
	A2	08	A0	90	00065		MOVB	8(R0),25(R2)		
	50	00000000G	EF	D0	0006A		MOVL	FDL_BLOCK,R0	:	2971
1E	A2	14	A0	90	00071		MOVB	20(R0),30(R2)		
00V00000001G	EF		00	E0	00076		BBS	#0,TEMP_FDL3\$TYPE+1,11\$:	2976
00V00000000G	EF		09	E1	0007E		BBC	#9,TEMP_FDL3\$TYPE,13\$		
00000000G	EF	00000000G	EF	7D	00086	11\$:	MOVQ	NULL_STRING,TEMP_DESCRIPTOR	:	2980
	50	00000000G	EF	D0	00091		MOVL	FDL_BLOCK,R0	:	2981
		6C	A0	9F	00098		PUSHAB	108(R0)		
	51	00000000G	EF	D0	0009B		MOVL	FDL_BLOCK,R1		
		68	A1	9F	000A2		PUSHAB	104(R1)		
04C1	CF		02	FB	000A5		CALLS	#2,CVT_QUAD_DESC		
00000000G	EF		50	7D	000AA		MOVQ	R0,TEMP_DESCRIPTOR		
		09	A2	9F	000B1		PUSHAB	9(R2)	:	2985
00000000G	EF	00000000G	EF	9F	000B4		PUSHAB	TEMP_DESCRIPTOR		
00V00000000G	EF		02	FB	000BA		CALLS	#2,LTB\$SCOPY_DXDX		
	OF	19	05	E1	000C1	13\$:	BBC	#5,TEMP_FDL3\$TYPE,15\$:	2992
			A2	91	000C9		CMPB	25(R2),#15		
00V00000000G	EF		00V	13	000CD		BEQL	19\$		
	21	1E	05	E0	000CF	15\$:	BBS	#5,TEMP_FDL3\$TYPE,17\$		
			A2	91	000D7		CMPB	30(R2),#33		
00V00000000G	EF		00V	13	000DB		BEQL	19\$		
	50	1E	05	E0	000DD	17\$:	BBS	#5,TEMP_FDL3\$TYPE,21\$		
	50		A2	9A	000E5		MOVZBL	30(R2),R0		
00V00000000G	EF		04	C4	000E9		MULL2	#4,R0		
00000000G	EF	00000000G	50	E1	000EC		BBC	R0,SEC_TYPE,21\$		
	50	00000000G	EF	7D	000F4	19\$:	MOVQ	NULL_STRING,TEMP_DESCRIPTOR	:	3008
		64	EF	D0	000FF		MOVL	FDL_BLOCK,R0	:	3009
	51	00000000G	A0	9F	00106		PUSHAB	100(R0)		
		60	EF	D0	00109		MOVL	FDL_BLOCK,R1		
04C1	CF		A1	9F	00110		PUSHAB	96(R1)		
00000000G	EF		02	FB	00113		CALLS	#2,CVT_QUAD_DESC		
		11	50	7D	00118		MOVQ	R0,TEMP_DESCRIPTOR		
		00000000G	A2	9F	0011F		PUSHAB	17(R2)	:	3013
	EF		EF	9F	00122		PUSHAB	TEMP_DESCRIPTOR		
00000000G	EF		02	FB	00128		CALLS	#2,LTB\$SCOPY_DXDX		
	50	19	A2	9A	0012F	21\$:	MOVZBL	25(R2),R0	:	3020
	10		50	D1	00133		CMPB	R0,#16		
00VFFFEB31	EF		00V	1E	00136		BGEQU	23\$		
	50	00000000G	50	E1	00138		BBC	R0,C.AAH,23\$		
1A	A2	0C	EF	D0	00140		MOVL	FDL_BLOCK,R0	:	3022
		1A	A0	D0	00147		MOVL	12(R0),26(R2)		
		1E	00V	11	0014C	23\$:	BRB	24\$		
	50		A2	D4	0014E	24\$:	CLRL	26(R2)	:	3026
00000098	8F		A2	9A	00151		MOVZBL	30(R2),R0	:	3028
			50	D1	00155		CMPB	R0,#152		
00VFFFEB0F	EF		00V	1E	0015C		BGEQU	26\$		
	50	00000000G	50	E1	0015E		BBC	R0,C.AAI,26\$		
1F	A2	18	EF	D0	00166		MOVL	FDL_BLOCK,R0	:	3030
			A0	D0	0016D		MOVL	24(R0),31(R2)		
87	8F	1E	00V	11	00172	26\$:	BRB	30\$		
			A2	91	00174		CMPB	30(R2),#-121	:	3037
1F	A2		00V	12	00179		BNEQ	28\$		
			07	D0	0017B		MOVL	#7,31(R2)	:	3039
			00V	11	0017F		BRB	29\$		
		1F	A2	D4	00181	28\$:	CLRL	31(R2)	:	3043
					00184	29\$:				

59	8F	1E	A2	91	00184	30\$:	CMPB	30(R2),#89		: 3063
			00V	13	00189		BEQL	32\$		
83	8F	1E	A2	91	0018B		CMPB	30(R2),#-125		
			00V	12	00190		BNEQ	33\$		
		23	A2	D4	00192	32\$:	CLRL	35(R2)		: 3074
	50	000000	00G	EF	D0	00195	MOVL	FDL BLOCK,R0		: 3075
27	A2	34	A0	D0	0019C		MOVL	52(R0),39(R2)		
			0000V	31	001A1		BRW	63\$		
29	50	1E	A2	9A	001A4	33\$:	MOVZBL	30(R2),R0		: 3086
62	8F		50	8F	001A8		CASEB	R0,#98,#41		
			0000V		001AD		.DISPL	34\$		
			0054		001AF		.DISPL	84		
			0054		001B1		.DISPL	84		
			0054		001B3		.DISPL	84		
			0054		001B5		.DISPL	84		
			0054		001B7		.DISPL	84		
			0054		001B9		.DISPL	84		
			0054		001BB		.DISPL	84		
			0054		001BD		.DISPL	84		
			0054		001BF		.DISPL	84		
			0054		001C1		.DISPL	84		
			0054		001C3		.DISPL	84		
			0054		001C5		.DISPL	84		
			0054		001C7		.DISPL	84		
			0054		001C9		.DISPL	84		
			0054		001CB		.DISPL	84		
			0054		001CD		.DISPL	84		
			0054		001CF		.DISPL	84		
			0054		001D1		.DISPL	84		
			0054		001D3		.DISPL	84		
			0054		001D5		.DISPL	84		
			0054		001D7		.DISPL	84		
			0054		001D9		.DISPL	84		
			0054		001DB		.DISPL	84		
			0054		001DD		.DISPL	84		
			0054		001DF		.DISPL	84		
			0054		001E1		.DISPL	84		
			0054		001E3		.DISPL	84		
			0054		001E5		.DISPL	84		
			0054		001E7		.DISPL	84		
			0054		001E9		.DISPL	84		
			0054		001EB		.DISPL	84		
			0054		001ED		.DISPL	84		
			0054		001EF		.DISPL	84		
			0054		001F1		.DISPL	84		
			0054		001F3		.DISPL	84		
			0054		001F5		.DISPL	84		
			0000V		001F7		.DISPL	50\$		
			0054		001F9		.DISPL	84		
			0054		001FB		.DISPL	84		
			0054		001FD		.DISPL	84		
			0000V		001FF		.DISPL	40\$		
			0000V	31	00201		BRW	61\$		
20	50	000000	00G	EF	D0	00204	34\$:	MOVL	FDL BLOCK,R0	: 3090
	00		34	A0	CF	0020B		CASEL	52(R0),#0,#32	
			0000V			00210		.DISPL	37\$	
			0042			00212		.DISPL	66	

		0042		00214		.DISPL	66		
		0042		00216		.DISPL	66		
		0042		00218		.DISPL	66		
		0042		0021A		.DISPL	66		
		0042		0021C		.DISPL	66		
		0042		0021E		.DISPL	66		
		0042		00220		.DISPL	66		
		0042		00222		.DISPL	66		
		0042		00224		.DISPL	66		
		0042		00226		.DISPL	66		
		0042		00228		.DISPL	66		
		0042		0022A		.DISPL	66		
		0042		0022C		.DISPL	66		
		0042		0022E		.DISPL	66		
		0000V		00230		.DISPL	36\$		
		0042		00232		.DISPL	66		
		0042		00234		.DISPL	66		
		0042		00236		.DISPL	66		
		0042		00238		.DISPL	66		
		0042		0023A		.DISPL	66		
		0042		0023C		.DISPL	66		
		0042		0023E		.DISPL	66		
		0042		00240		.DISPL	66		
		0042		00242		.DISPL	66		
		0042		00244		.DISPL	66		
		0042		00246		.DISPL	66		
		0042		00248		.DISPL	66		
		0042		0024A		.DISPL	66		
		0042		0024C		.DISPL	66		
		0042		0024E		.DISPL	66		
		0000V		00250		.DISPL	35\$		
		00V	11	00252		BRB	38\$		
23	A2	1F	D0	00254	35\$:	MOVL	#31,35(R2)		: 3092
		0000V	31	00258		BRW	62\$		
23	A2	1E	D0	0025B	36\$:	MOVL	#30,35(R2)		: 3093
		0000V	31	0025F		BRW	62\$		
23	A2	1D	D0	00262	37\$:	MOVL	#29,35(R2)		: 3094
		0000V	31	00266		BRW	62\$		
		0000V	31	00269	38\$:	BRW	62\$		
	50	00000000G	EF	D0	0026C	40\$:	MOVL	FDL BLOCK,R0	: 3104
06	00	34	AO	CF	00273	CASEL	52(R0),#0,#6		
		0000V		00278		.DISPL	45\$		
		0000V		0027A		.DISPL	41\$		
		0000V		0027C		.DISPL	46\$		
		0000V		0027E		.DISPL	47\$		
		0000V		00280		.DISPL	42\$		
		0000V		00282		.DISPL	44\$		
		0000V		00284		.DISPL	43\$		
		00V	11	00286		BRB	48\$		
23	A2	0D	D0	00288	41\$:	MOVL	#13,35(R2)		: 3106
		00V	11	0028C		BRB	49\$		
23	A2	10	D0	0028E	42\$:	MOVL	#16,35(R2)		: 3107
		00V	11	00292		BRB	49\$		
23	A2	12	D0	00294	43\$:	MOVL	#18,35(R2)		: 3108
		00V	11	00298		BRB	49\$		
23	A2	11	D0	0029A	44\$:	MOVL	#17,35(R2)		: 3109
		00V	11	0029E		BRB	49\$		

Generated Code								
23	A2	0C	DO	002A0	45\$:	MOVL	#12,35(R2)	; 3110
		00V	11	002A4		BRB	49\$	
23	A2	0E	DO	002A6	46\$:	MOVL	#14,35(R2)	; 3111
		00V	11	002AA		BRB	49\$	
23	A2	0F	DO	002AC	47\$:	MOVL	#15,35(R2)	; 3112
		00V	11	002B0		BRB	49\$	
				002B2	48\$:			
		00V	11	002B2	49\$:	BRB	62\$	
07	50	00000000G	EF	DO	002B4	50\$:	MOVL	FDL BLOCK,R0
	00	34	AO	CF	002BB		CASEL	52(R0),#0,#7
		0000V		002C0		.DISPL	58\$	
		0000V		002C2		.DISPL	55\$	
		0000V		002C4		.DISPL	51\$	
		0000V		002C6		.DISPL	56\$	
		0000V		002C8		.DISPL	52\$	
		0000V		002CA		.DISPL	54\$	
		0000V		002CC		.DISPL	57\$	
		0000V		002CE		.DISPL	53\$	
		00V	11	002D0		BRB	59\$	
23	A2	23	DO	002D2	51\$:	MOVL	#35,35(R2)	; 3124
		00V	11	002D6		BRB	62\$	
23	A2	25	DO	002D8	52\$:	MOVL	#37,35(R2)	; 3125
		00V	11	002DC		BRB	62\$	
23	A2	27	DO	002DE	53\$:	MOVL	#39,35(R2)	; 3126
		00V	11	002E2		BRB	62\$	
23	A2	28	DO	002E4	54\$:	MOVL	#40,35(R2)	; 3127
		00V	11	002E8		BRB	62\$	
23	A2	22	DO	002EA	55\$:	MOVL	#34,35(R2)	; 3128
		00V	11	002EE		BRB	62\$	
23	A2	24	DO	002F0	56\$:	MOVL	#36,35(R2)	; 3129
		00V	11	002F4		BRB	62\$	
23	A2	26	DO	002F6	57\$:	MOVL	#38,35(R2)	; 3130
		00V	11	002FA		BRB	62\$	
23	A2	21	DO	002FC	58\$:	MOVL	#33,35(R2)	; 3131
		00V	11	00300		BRB	62\$	
		00V	11	00302	59\$:	BRB	62\$	
	50	00000000G	EF	DO	00304	61\$:	MOVL	FDL BLOCK,R0
23	A2	34	AO	DO	0030B		MOVL	52(R0),35(R2)
	50	00000000G	EF	DO	00310	62\$:	MOVL	FDL BLOCK,R0
27	A2	38	AO	DO	00317		MOVL	56(R0),39(R2)
	50	00000000G	EF	DO	0031C	63\$:	MOVL	FDL BLOCK,R0
	00V	3C	AO	E9	00323		BLBC	60(R0),65\$
28	A2		01	90	00327		MOVB	#1,43(R2)
			00V	11	0032B		BRB	66\$
		28	A2	94	0032D	65\$:	CLRB	43(R2)
	50	00000000G	EF	DO	00330	66\$:	MOVL	FDL BLOCK,R0
2C	A2	40	AO	DO	00337		MOVL	64(R0),44(R2)
	50	00000000G	EF	DO	0033C		MOVL	FDL BLOCK,R0
30	A2	48	AO	DO	00343		MOVL	72(R0),48(R2)
	50	00000000G	EF	DO	00348		MOVL	FDL BLOCK,R0
34	A2	4C	AO	DO	0034F		MOVL	76(R0),52(R2)
	50	00000000G	EF	DO	00354		MOVL	FDL BLOCK,R0
38	A2	50	AO	DO	0035B		MOVL	80(R0),56(R2)
	50	00000000G	EF	DO	00360		MOVL	FDL BLOCK,R0
3C	A2	54	AO	DO	00367		MOVL	84(R0),60(R2)
	50	1E	A2	9A	0036C		MOVZBL	30(R2),R0
00000098	8F		50	D1	00370		CMPL	R0,#152

00VFFFE908	EF		00V	1E	00377	BGEQU	67\$	
00V00000000G	EF		50	E0	00379	BBS	R0,C.AAJ,71\$	
	50	19	00	E1	00381	BBC	#0,ANALYSIS_ONLY,69\$	
	10		A2	9A	00389	MOVZBL	25(R2),R0	
			50	D1	0038D	CMPL	R0,#16	
			00V	1E	00390	BGEQU	69\$	
00VFFFE903	EF		50	E0	00392	BBS	R0,C.AAK,71\$	
00V00000000G	EF		00	E0	0039A	BBS	#0,ANALYSIS_ONLY,75\$	
	50	19	A2	9A	003A2	MOVZBL	25(R2),R0	
	10		50	D1	003A6	CMPL	R0,#16	
			00V	1E	003A9	BGEQU	71\$	
00VFFFE8EC	EF		50	E0	003AB	BBS	R0,C.AAL,75\$	
	02		62	91	003B3	CMPL	(R2),#2	: 3181
			00V	12	003B6	BNEQ	73\$	
00000000G	EF	00000000G	EF	D0	003B8	MOVL	DEF_TAIL,DEF_CURRENT	: 3185
ODED	CF		00	FB	003C3	CALLS	#0,INSERT_AFTER_CURRENT	: 3186
			00V	11	003C8	BRB	74\$	
		00000001	8F	DF	003CA	PUSHAL	#1	: 3192
1189	CF		01	FB	003D0	CALLS	#1,INSERT_IN_ORDER	
					003D5			
					003D5			
	50		5C	D0	003D5	MOVL	EDF\$LINE_PARSED,R0	: 3196
			04	003D8	RET			

; Routine Size: 985 bytes, Routine Base: \$CODE + 01413

017EC

.END

COMMAND QUALIFIERS

PASCAL/MACHINE/NODEBUG/NOCHECK/LIS=LISS:EDFUTIL/OBJ=OBJ\$:EDFUTIL MSRC\$:EDFUTIL

/CHECK=(NOBOUNDS,NOCASE_SELECTORS,NOOVERFLOW,NOPOINTERS,NOSUBRANGE)

/DEBUG=(NOSYMBOL\$,NOTRACEBACK)

/ENVIRONMENT= \$255\$DUA28:[EDF.OBJ]EDFUTIL.PEN;1

/LIST= \$255\$DUA28:[EDF.LIS]EDFUTIL.LIS;1

/OBJECT= \$255\$DUA28:[EDF.OBJ]EDFUTIL.OBJ;1

/NOCROSS_REFERENCE /ERROR_LIMIT=30 /NOG_FLOATING /MACHINE_CODE /NOOLD_VERSION /OPTIMIZE /NOSTANDARD /WARNINGS

COMPILER INTERNAL TIMING

Phase	Faults	CPU Time	Elapsed Time
Initialization	91	00:00.5	00:03.5
Source Analysis	972	00:21.5	03:10.5
Source Listing	50	00:03.9	00:07.8
Tree Construction	134	00:01.6	00:03.5
Flow Analysis	46	00:00.9	00:02.1
Profit Analysis	51	00:01.2	00:02.9
Context Analysis	479	00:13.8	00:26.7
Name Packing	6	00:00.4	00:00.9
Code Selection	41	00:01.9	00:03.6
Final	437	00:07.0	00:17.4
TOTAL	2311	00:52.9	04:19.1

COMPILATION STATISTICS

CPU Time:	00:52.9	(3627 Lines/Minute)
Elapsed Time:	04:19.1	
Page Faults:	2311	
Compilation Complete		

0128 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0129 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY